



ENOVIA SmarTeam | Dassault Systèmes
www.smartteam.com
www.3ds.com

ENOVIA SMARTTEAM

SMARTIXF LIBRARY

Programmer's Guide

Important Notice

© Dassault Systèmes, 2004, 2008. All rights reserved.

CATIA, ENOVIA, SMARTEAM and the 3DS logo are registered trademarks of Dassault Systèmes or its subsidiaries in the US and/or other countries.

PROPRIETARY RIGHTS NOTICE: This documentation is the property of Dassault Systèmes. This documentation shall be treated as confidential information and may only be used by employees or contractors of the Customer in accordance with the terms of the End-User License Agreement accepted by Customer.

Any use of the Licensed Program contained in this media or accompanying it, is subject to the terms of the End User License Agreement accepted by Customer. The Licensed Program is protected by international copyright laws and international treaties. Unauthorized use, reproduction and/or distribution of any of the Licensed Program, or any part thereof, may result in severe civil and/or criminal penalties, and will be prosecuted to the maximum extent possible under the law. Company names and product names mentioned herein are the property of their respective owners and certain portions of the Licensed Program contain elements subject to copyright owned by these entities. See the Documentation CD provided with the Licensed Program for details and/or additional terms and conditions relating to these entities.

Part Number: DVS-A2-180007

Table of Contents

1	INTRODUCTION	1
	NAMING CONVENTIONS	1
	<i>NCName</i>	1
	<i>Class Behavior URI</i>	1
2	OVERVIEW OF OBJECTS.....	2
	ISMIXFSHEMA	2
	<i>ISmIxfClassesBehaviors</i>	4
	<i>ISmIxfClasses</i>	8
	<i>ISmIxfDomainBehaviors</i>	17
	<i>ISmIxfInfo</i>	19
	<i>Common Tasks</i>	22
	SMIXFINITIALIZATIONDATA	25
	SMIXFWRITER	27
	<i>ISmIxfDataWriter</i>	31
	<i>ISmIxfSchema</i>	36
	<i>Common Tasks</i>	36
	SMIXFREADER	40
	<i>ISmIxfDataReader</i>	41
	<i>ISmIxfUnderstoodInfoItems</i>	44
	<i>ISmIxfSchema</i>	44
	<i>Common Tasks</i>	45
	READING AND WRITING AN EXTERNAL SCHEMA	46
	<i>SmIxfExternalSchemaWriter</i>	46
	<i>SmIxfExternalSchemaReader</i>	46
	ISMIXFSTDHELPER	47
	<i>Standard Behaviors</i>	47
	<i>ISmIxfSchemaHelper</i>	48
	<i>ISmIxfWriterHelper</i>	56
	<i>ISmIxfReaderHelper</i>	79
3	SAMPLE IXF APPLICATION.....	88
	<i>Messaging Format</i>	89
	<i>Class Behaviors</i>	90
	<i>Domain Behaviors</i>	90
	<i>Connectivity of Objects</i>	92
	IMPLEMENTING THE APPLICATION	94
	<i>Creating the Schema</i>	94
	<i>Writing the Data</i>	100
	<i>Reading the Data</i>	105
	<i>Executing the Application</i>	107

1 Introduction

The **IXF** library enables you to perform the following functions:

- Generating an iXF schema
- Processing an iXF schema
- Generating an iXF Archive File
- Processing an iXF Archive File

The iXF format created and processed by the iXF library conforms to the format described in the “iXF Specifications 1.0” document, available at <http://www.ixfstd.org/std/docs/ixf>.

For additional information on the iXF format, see the iXF Standard web site, at <http://www.ixfstd.org>.

Naming Conventions

This section describes the naming conventions used in this guide.

NCName

A valid NCName must begin with a letter or an underscore (`_`) and cannot contain spaces; letters, digits, and underscores are allowed after the first character:

NCName ::= ([Letter](#) | ' `_` ') ([NCNameChar](#))

NCNameChar ::= [Letter](#) | [Digit](#) | ' `.` ' | ' `-` ' | ' `_` '

For example: “PartMaster”. See also <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

Class Behavior URI

A valid class behavior URI must contain a namespace and a behavior name, separated by “#” as: Class Behavior URI ::= (Namespace) (‘#’) (Name).

For example,
“<http://www.ixfstd.org/std/ns/core/classBehaviors/links/1.0#link>”

2 Overview of Objects

This chapter presents an overview of the main SmartIxf objects including a description of the associated objects that are useful for the programmer:

- ISmIxfSchema Object
- SmIxfWriter Object
- SmIxfReader Object
- ISmIxfStdHelper Object

ISmIxfSchema

The ISmIxfSchema object serves to organize and refer to the components of the iXF schema document.

The Schema is maintained wholly in memory. The Schema can be saved to a file in two ways:

- Through SmIxfExternalSchemaWriter
- When creating an archive file using the SmIxfWriter.

The ISmIxfSchema object is not a top-level object in the object hierarchy but is contained in the following top-level objects:

SmIxfWriter– for writing the iXF schema document

SmIxfReader – for reading the iXF schema document

SmIxfExternalSchemaWriter – for writing an external iXF schema document.

SmIxfExternalSchemaReader– for reading an external iXF schema document

Because of its importance, the ISmIxfSchema object is discussed separately in this major section; refer to the sections for the above objects for information on how the ISmIxfSchema object is included in those objects.

The schema components and their corresponding objects are shown in the following object diagram:

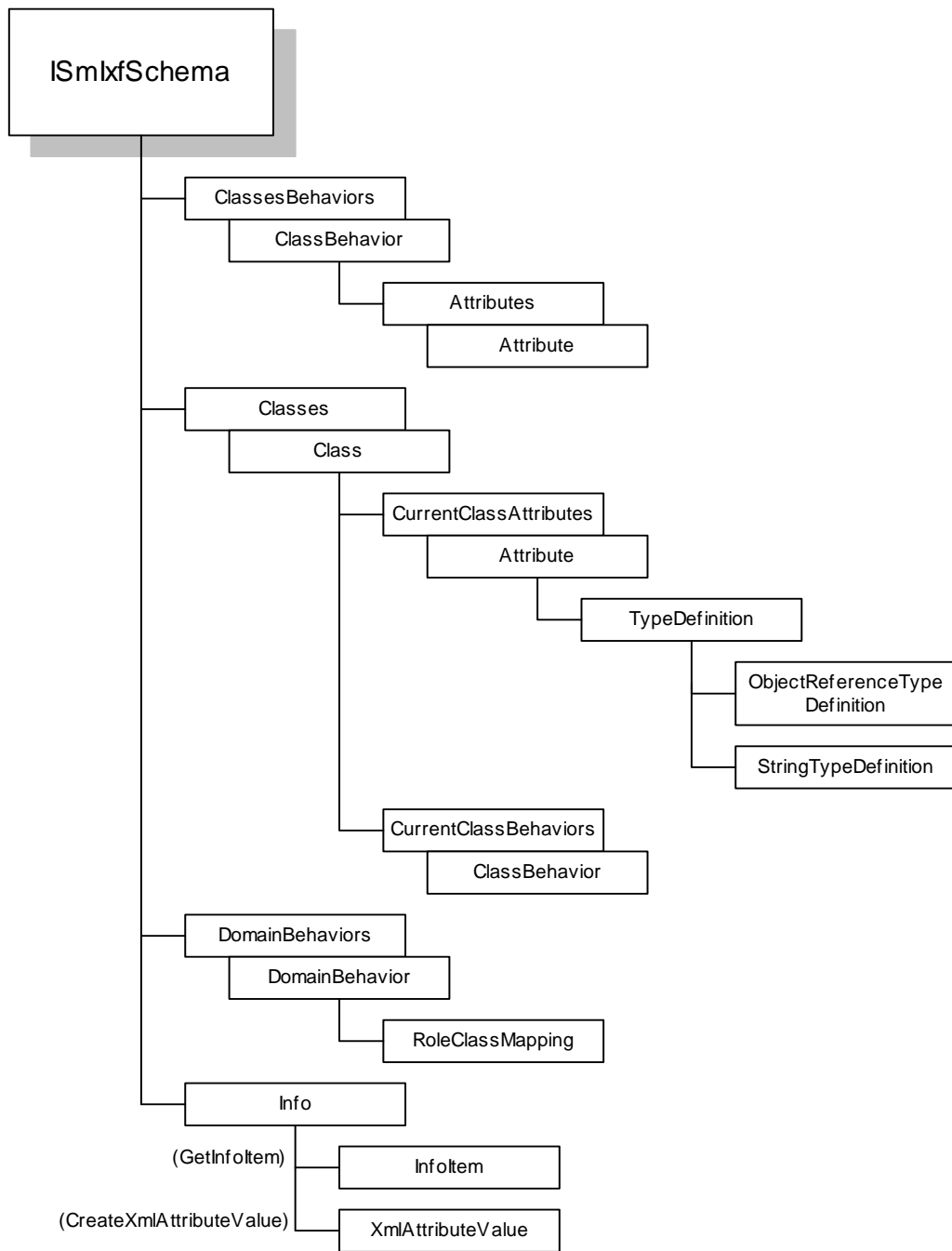


Figure 2-1 ISmIxfSchema Object Diagram

Properties

The ISmIxfSchema object has the following properties

Property	Description
Classes	Collection ISmIxfClasses of schema classes.
ClassesBehaviors	Collection ISmIxfClassesBehaviors of schema ClassBehaviors
DomainBehaviors	Collection ISmIxfDomainBehaviors of schema DomainBehaviors
Info	ISmIxfInfo object for holding miscellaneous information
SchemaLocation	The physical location of the schema file
SchemaURI	The Schema URI, which is the unique identifier of the schema.

Obtaining the ISmIxfSchema Object

To create an ISmIxfSchema Object from the SmIxfWriter Object (to create a SmIxfWriter object see SmIxfWriter):

```
IxfWriter.Schema
```

A Schema object can also be obtained similarly from the SmIxfReader Object, SmIxfExternalSchemaWriter Object, and from the SmIxfExternalSchemaReader Object.

ISmIxfClassesBehaviors

An ISmIxfClassesBehaviors object is a collection of ISmIxfClassBehavior objects and represents all ISmIxfClassBehavior objects related to the IXF schema.

Note: This object is not the same as the ISmIxfClassBehaviors, which represents all ISmIxfClassBehavior objects declared by a specific class.

Adding a New ClassBehavior to the IXF Schema

You use the Add method of the ClassesBehaviors object to add a new ClassBehavior object to the IXF Schema. Once you have added a ClassBehavior to the ClassesBehaviors object you can declare it in a specific class. Before using the Add method, you need to understand how a ClassBehavior is packaged.

ClassBehavior Schema File

A ClassBehavior object is defined in a ClassBehavior schema file. The ClassBehavior schema file can be packaged either embedded in or external to the IXF Archive file as described below.

ClassBehavior Schema File Packaging State

The following table describes the packaging states of the ClassBehavior schema file and the software constant used for each state. The packaging state of the ClassBehavior schema file is described by the ModeTypeEnum parameter of the Add function.

Packaging State	Description	ModeTypeEnum Software Constant
Embedded	The class behavior definition is created and saved to a ClassBehavior schema file, which is embedded in the iXF archive file. The ClassBehavior schema is specified by the namespace of the behavior.	mtEmbedded
External	The class behavior was previously defined and the definition is in a schema file. The ClassBehavior definition is taken from this external ClassBehavior schema file. The ClassBehavior schema file is not embedded in the iXF package; it is specified by its BehaviorURI.	mtExternal

Add Method

The Add method is called as follows:

```
Set ClassBehavior = Schema.ClassesBehaviors.Add(ModeTypeEnum, BehaviorURI,  
[SchemaLocation], [Load=false])
```

The arguments of the method are:

Argument	Description
Mode	Packaging state of the schema – one of ModeTypeEnum. See table above.
BehaviorURI	The Class Behavior URI
SchemaLocation	The behavior schema physical location. Specified only when the Mode is mtExternal.
Load	Whether or not the behavior schema needs to be loaded. The default is false. Can be set to true only when the Mode argument is mtExternal.

Example

```
Dim IxfClassBehavior as ISmIxfClassBehavior  
'Create a Class Behavior "link"  
Set IxfClassBehavior = Schema.ClassesBehaviors.Add(  
mtEmbedded, "http://www.ixfstd.org/std/ns/core/classBehaviors/links/1.0#link")
```

See ISmIxfClassBehavior for more information on that object.

ISmIxfClassBehavior

A ClassBehavior lets you define a set of class attributes as an entity separate from any specific class, where the entity is identified by a unique URI reference. A ClassBehavior so defined becomes a standard set of attributes, which can be “implemented” by one or more classes, as required. In this way, a ClassBehavior is similar to an Interface of a programming language like JavaTM; any class that wants to implement a ClassBehavior needs to declare the ClassBehavior usage.

A ClassBehavior is defined in the schema separately from the class definitions and is used in a specific class by declaring it in the class definition in the schema (see Adding a ClassBehavior to a Class in [ISmIxfClassBehaviors](#).) The attribute values for attributes in the ClassBehavior are assigned in the

instantiation of the class in the data file, in the same way that values are assigned to the internal attributes of the class.

[Figure 2-2](#) shows how ClassBehaviors are used in schema class definitions. It shows the internal attribute definitions of a class and the ClassBehavior declarations. You can declare the same ClassBehavior in more than one class, and you can declare more than one ClassBehavior in a single class.

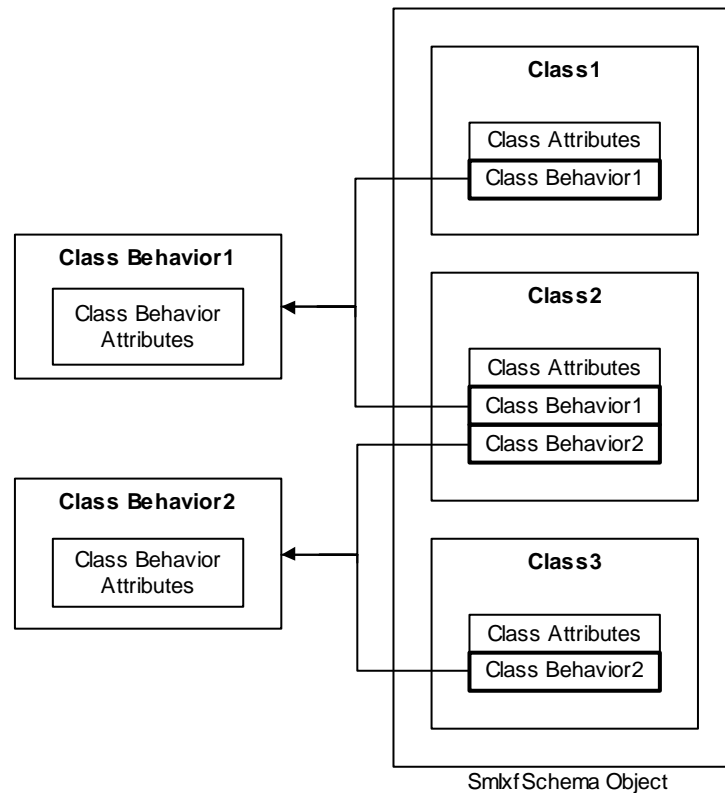


Figure 2-2 Class Behaviors

Properties

An ISmIxfClassBehavior object has the following properties:

Property	Description
Attributes	Collection ISmIxfAttributes of class behavior attributes
SchemaLocation	The physical location of the ClassBehavior schema file, in the event that the ClassBehavior was previously defined and saved to a schema file, the definition can be obtained (loaded) through this schema file. Otherwise, if the class behavior is defined in the schema (and not loaded) – the SchemaLocation parameter is an empty string.
URI	The Class Behavior URI, which is the unique identifier of the class behavior.

Adding an Attribute to a ClassBehavior

Use the Add method of the IxfClassBehavior.Attributes collection object to add an attribute to the IxfClassBehavior.

```
Set IxfAttribute = IxfClassBehavior.Attributes.Add(AttributeName)
```

The Add method returns an object of type ISmIxfAttribute. Specify the AttributeName as a valid NCName. For more information about Attributes, see ISmIxfAttribute.

For an example of how to add an attribute to a ClassBehavior, see Common Tasks, “ISmIxfSchema: Defining a ClassBehavior”.

ISmIxfClasses

An ISmIxfClasses object is a collection of ISmIxfClass objects.

Adding a Class to the IXF Schema

Use the Add method of the ISmIxfClasses object to add a class to the IXF Schema. The Add method returns an object of type ISmIxfClass. You specify the name of the class as a valid NCName.

The Add method is called as follows:

```
Set IxfClass = Schema.Classes.Add(ClassName)
```

Where ClassName has to be a valid NCName.

Once you have added the class you can specify the class properties, as in the next section.

For an example of how to add a class to the IXF Schema, see Common Tasks, “ISmIxfSchema: Creating a Schema with Classes and Class Attributes”.

ISmIxfClass

The following figure shows the object diagram for the ISmIxfClass Object.

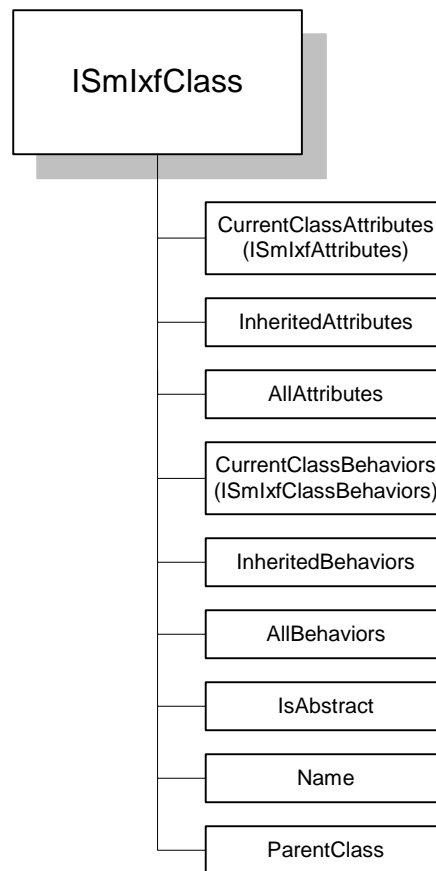


Figure 2-3 ISmIxfClass Object Diagram

Properties

An ISmIxfClass object has the following properties:

Property	Description
Name	The class name must be a valid NCName.
Parent Class	The parent class object
CurrentClassAttributes	Collection ISmIxfAttributes of the class attributes. Does not include the inherited class attributes.
InheritedAttributes	Collection ISmIxfReadOnlyAttributes of the inherited class attributes.
AllAttributes	Collection ISmIxfReadOnlyAttributes of all the class attributes, including inherited attributes.
CurrentClassBehaviors	Collection ISmIxfClassBehaviors of class behaviors that are supported by the class. Does not include the class behaviors that were inherited.
InheritedBehaviors	Collection ISmIxfReadOnlyClassBehaviors of the inherited class behaviors.
AllBehaviors	Collection ISmIxfReadOnlyClassBehaviors of all the class behaviors that are supported by the class, including inherited class behaviors.
IsAbstract	Indicates whether or not the class is abstract. If it is, it cannot be instantiated

ISmIxfAttributes

An ISmIxfAttributes object is a collection of ISmIxfAttribute objects.

The CurrentClassAttributes is a collection of ISmIxfAttributes objects, which represent the attributes defined internally to the current class. The ClassBehavior object also includes a collection ISmIxfAttributes, which represents the attributes of the ClassBehavior (see [ISmIxfClassBehaviors](#).)

Adding an Attribute to a Class

Use the Add method of the ISmIxfAttributes object to add an attribute to the collection. The Add method returns an object of type ISmIxfAttribute. It is called as follows:

```
Set IxfAttribute = IxfClass.CurrentClassAttributes.Add(AttributeName)
```

Where `AttributeName` must be a valid `NCName`.

Once you have added the attribute you can specify the attribute properties.

For an example of how to add an attribute to `CurrentClassAttributes`, see Common Tasks, “ISmIxfSchema: Creating a Schema with Classes and Class Attributes”.

ISmIxfAttribute

The `ISmIxfAttribute` object represents an individual class attribute or an individual attribute of a `ClassBehavior`.

Properties

The `ISmIxfAttribute` object has the following properties:

Property	Description
Name	The attribute name. A valid <code>NCName</code> .
Default value	The default value of the attribute. It is assigned as an object's (class or <code>ClassBehavior</code>) attribute value in case no value was assigned.
IsNullAllowed	True if the attribute value can be set to Null. Default is true
IsPrimary	True if the attribute is part of the class primary identifier. The default is false.
Required	True if the attribute is required. If it is, it has to be assigned, or a default value must be indicated in the Default value property. Default is false.
TypeDefinition	Returns the data type of the attribute. Returns an object of type <code>ISmIxfTypeDefinition</code> .

Note: `ISmIxfAttribute` is only the definition of the attribute structure; the actual value of this attribute is inserted by the `SmlxfWriter` object.

ISmIxfTypeDefinition

The `ISmIxfTypeDefinition` Object specifies the data type of the `ISmIxfAttribute` object.

Properties

The ISmIxfTypeDefinition object has three properties:

Property	Description
ValueType	The ValueType property is an Enum type DataTypeEnum that specifies the type of the value that can be assigned to the attribute. It is a subset of the W3C XML Schema Data Types, as defined in http://www.w3.org/TR/xmlschema-2/ . See Table 1 for a list of data types.
ObjectReferenceType	If ValueType is set to dtObjectReference, the ObjectReferenceType property lets you specify more information about the object reference. See below for more information.
StringType	If the ValueType is assigned to dtString then the StringType property lets you specify more information about the string. See below for more details.

Table 1 ValueType Data Types

ValueType	Description	Software Constant
String	Character strings in XML	dtString
Boolean	Binary-valued logic	dtBoolean
Float	IEEE single-precision 32-bit floating point type	dtFloat
Double	IEEE double-precision 64-bit floating point type	dtDouble
Duration	Duration of time	dtDuration
Base64Binary	Base64-encoded arbitrary binary data	dtBase64Binary
HexBinary	Arbitrary hex-encoded binary data	dtHexBinary
AnyUri	A Uniform Resource Identifier Reference (URI).	dtAnyUri

Language	Natural language identifiers as defined by [RFC 1766].	dtLanguage
Int	Integer between -2147483648 and 2147483647.	dtInt
Short	Integer between -32768 and 32767	dtShort
Byte	Integer between -128 and 127	dtByte
UnsignedShort	Integer between 0 and 65535	dtUnsignedShort
UnsignedByte	Integer between 0 and 255	dtUnsignedByte
DateTime	A specific instant of time	dtDateTime
Time	An instant of time that recurs every day	dtTime
Date	A calendar date	dtDate
gMonth	A gregorian month that recurs every year	dtGMonth
gYear	A gregorian calendar year	dtGYear
ObjectReference	An object	dtObjectReference
XML	XML text	dtXML

Depending on ValueType, there can be additional options, as discussed in the following sections.

Specifying Information about an Object Reference

If ISmIXfTypeDefinition.ValueType is set to dtObjectReference, the ObjectReferenceType property lets you specify more information about the object reference.

Properties

The ObjectReferenceType property returns an object of type ISmIXfObjectReferenceTypeDefinition, which has the properties:

Property	Description
----------	-------------

RestrictionType	You use this property to place restrictions on the type of object referenced through the ObjectReferenceType property. Set to one of ObjectReferenceRestrictionTypeEnum.
ClassName	Specifies a class to which the objects referenced or their descendants must belong. Can be accessed only if RestrictionType has the value ortClass or ortClassAndDescendants. See details of ObjectReferenceType Restrictions . RestrictionType below for more information.
BehaviorURI	Specifies a behavior that the object referenced must implement. Can be accessed only if the RestrictionType has the value ortBehavior. See details of ObjectReferenceType.RestrictioType below for more information.

ObjectReferenceType Restrictions

You use the RestrictionType property to place restrictions on the type of object that can be referenced through the ObjectReferenceType property. This helps you tailor an attribute for special use.

The RestrictionType property is available as follows:

`IxfAttribute.TypeDefinition.ObjectReferenceType.RestrictioType`

The RestrictionType can take one of the following ObjectReferenceRestrictionTypeEnum values:

RestrictionType	Description	Software Constant
Any	The reference can be to any kind of object (the default)	ortAny
Class	The reference can be to an object of a specific class only. The class name should be assigned to TypeDefinition.ObjectReferenceType.ClassName (continued).	ortClass
ClassAnd Descendants	The reference can be to an object of a specific class or its descendants only. The class name should be assigned to TypeDefinition.ObjectReferenceType.ClassName (continued).	OrtClassAnd Descendants
Behavior	The reference can be to an object that implements a specific behavior only. The behavior URI should be assigned to TypeDefinition.ObjectReferenceType.BehaviorURI (continued).	ortBehavior

Example

The following code allows the attribute to reference only objects of the class “DocumentMaster”.

```

IxfAttribute.TypeDefinition.ValueType = dtObjectReference
IxfAttribute.TypeDefinition.ObjectReferenceType.RestrictionType = ortClass
IxfAttribute.TypeDefinition.ObjectReference.ClassName = "DocumentMaster"

```

String Type Options

If the `ISmIxfTypeDefinition.ValueType` is assigned to `dtString` then you can specify a maximum length for the string by `ISmIxfTypeDefinition.StringType.MaxLength`. The default for `MaxLength` is 0, which means there is no restriction for the string length.

Example

The following code restricts the length of the string attribute to 50 characters.

```
IxfAttribute.TypeDefinition.ValueType = dtString  
IxfAttribute.TypeDefinition.StringType.MaxLength = 50
```

ISmIxfClassBehaviors

The `ISmIxfClassBehaviors` Object is a collection of `ISmIxfClassBehavior` objects. It represents the set of `ClassBehaviors` declared in a specific class.

Note: This object is not the same as the collection object [ISmIxfClassesBehaviors](#). The latter refers to the set of all `ClassBehavior` objects associated with the entire schema and defined externally to all classes.

Adding a ClassBehavior to a Class

Once you have added a `ClassBehavior` to the `ClassesBehaviors` object (see [Adding a New ClassBehavior to the IXF Schema](#)), you can declare the `ClassBehavior` in a specific class. You use the `Add` method of the `ClassBehaviors` object to add a `ClassBehavior` object to `CurrentClassBehaviors`.

The `Add` function is called as follows:

```
IxfClass.CurrentClassBehaviors.Add(Behavior, MustUnderstandEnum)
```

The method parameters are as follows:

Parameter	Description
Behavior	A ClassBehavior object that already exists in the collection Schema.ClassesBehaviors.
MustUnderstand	Denotes whether this Class Behavior, when declared in this class, must be understood by the reading processor. Possible values: muYes, muNo

For an example, see Common Tasks, ISmIxfSchema: Declaring usage of a class behavior by a class.

ISmIxfDomainBehaviors

An ISmIxfDomainBehaviors object is a collection of ISmIxfDomainBehavior objects.

Adding a DomainBehavior to DomainBehaviors

You use the Add method of the DomainBehaviors object to add a DomainBehavior object to DomainBehaviors.

The Add function is called as follows:

```
Add(URI)
```

ISmIxfDomainBehavior

Conceptually, a Domain Behavior is composed of sets of Class Behaviors called Roles, where the Domain Behavior also specifies the classes that declare the Class Behaviors for each Role.

Specifically, a Domain Behavior defines a set of Roles and a set of Role-to-Class mappings (see Section 2.5 of the IXF Specification.) Each Role is associated with a set of Class Behaviors, which are specified by the documentation describing the Domain Behavior. The class that is mapped to the Role according to its Role-to-Class Mapping must declare the Role's Class Behaviors.

Note: It is very important to make sure that a class is mapped to a Role only if it implements the required class behaviors, even though this is not currently enforced by the API. The required class behaviors can be verified by consulting the Domain Behavior documentation.

Figure 2-4 shows the relationship between a Domain Behavior definition and a schema that uses it:

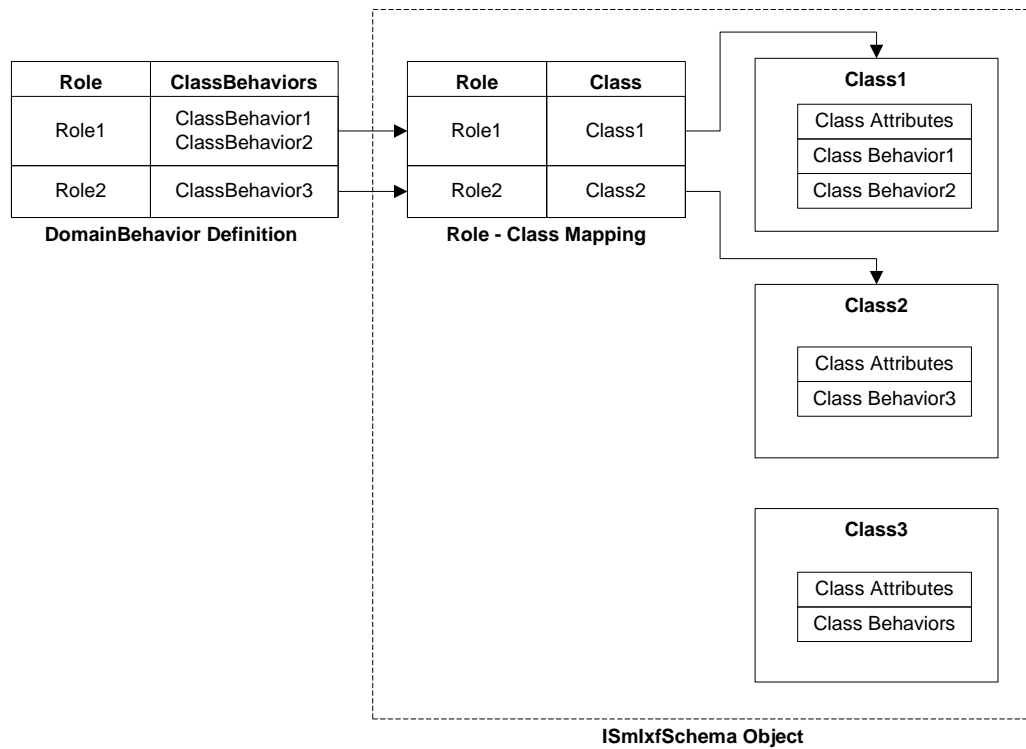


Figure 2-4 Domain Behavior

Properties

An ISmIxfDomainBehavior object has the following properties:

Property	Description
URI	The unique identifier of the DomainBehavior
RoleClassMapping	The RoleClassMapping property defines an association between role names and ISmIxfClass objects, whereby each role is assigned to a class.

ISmIxfInfo

The ISmIxfInfo Object holds miscellaneous information, which cannot be categorized as classes or behaviors. It is a collection of ISmIxfInfoItem objects.

Methods

It has the methods:

Method	Description
GetInfoItem	Gets an InfoItem from the collection by Name and Namespace. If it doesn't exist, a new object is created and added to the collection.
Save	Not used when Info accessed through Schema.
CreateXmlAttributeValue	Creates an ISmIxfXmlAttributeValue object

The ISmIxfInfo Object is obtained through the ISmIxfSchema object, as follows:

```
Set IxfInfo = Schema.Info
```

Note: ISmIxfInfo Object can also be obtained through the IxfWriter.DataWriter and IxfReader.DataReader. When the schema is saved ISmIxfInfo is saved automatically with the schema; in the Writer it has to be saved with the Save function.

Adding an InfoItem to a Info Object

Use the GetInfoItem method of Info to add a new InfoItem to the Info collection.

```
Set IxfInfoItem = IxfInfo.GetInfoItem(Name, Namespace)
```

ISmIxfInfoItem

The ISmIxfInfoItem object represents a member of the ISmIxfInfo collection, that is, a basic unit of miscellaneous information in the schema file.

Properties

The ISmIxfInfoItem object has the properties:

Property	Description
Name	InfoItem name
Namespace	InfoItem namespace
Value	InfoItem value
ValueType	Value type (see Table 1)
MustUnderstand	MustUnderstand flag for this InfoItem. If set to true, the reading process stops when this InfoItem is not in the list SmIxfReader.ISmIxfUnderstoodInfoItems, and Reader.ValidateMustUnderstand = true.

Use the GetInfoItem (Name, Namespace) method of the ISmIxfInfo Object to create an ISmIxfInfoItem object or to get an existing one.

For an example of how to add miscellaneous information to a ClassBehavior, see Common Tasks, ISmIxfSchema: Adding miscellaneous information.

Note: The ISmIxfInfo object under DataWriter represents another, independent way to write miscellaneous data, which you can use instead of or in addition to this ISmIxfInfo object under the ISmIxfSchema object. The difference is that with the current ISmIxfInfo object you do not need to save the object; it is saved automatically with the Schema.

ISmIxfXmlAttributeValue

The ISmIxfXmlAttributeValue object represents the value of an InfoItem of type “dtXML”. This object lets you insert miscellaneous information in the form of XML text. The XML text does not need to be a complete XML document, but it must be valid and well formed.

If the meaning of a prefix is not included in the XML text, you can provide it in the Namespaces property.

To create an ISmIxfXmlAttributeValue object, use the ISmIxfInfo method CreateXmlAttributeValue, as follows:

```
Set IxfXmlAttributeValue = Info.CreateXmlAttributeValue
```

Properties

The ISmIxfXmlAttributeValue object has the properties:

Property	Description
Namespaces	Returns an object ISmIxfNamespaces, a list of mappings between prefix and namespace that represents the meaning of each prefix that occurs in the XML string.
XML	A well-formed valid XML text as a WideString.

Note: The ISmIxfXmlAttributeValue object can also be used to provide an Xml text attribute value to a class attribute, when using the Writer object. See ISmIxfAttributesValues for more information.

Note: The XML string might be changed by the API but the meaning will stay the same.

Example

```
Dim Info As ISmIxfInfo
Dim InfoItem As ISmIxfInfoItem
Dim NameSpaces As ISmIxfNamespaces
Dim XmlAttributeValue As ISmIxfXmlAttributeValue
Dim XmlText As String

Set Info = Schema.Info
InfoItem = Info.GetInfoItem("XmlText", "http://...")
InfoItem.ValueType = dtXml
Set XmlAttributeValue = Info.CreateXmlAttributeValue
XmlText = "<p:name>John Bryce<p:name>"
XmlAttributeValue.XML = XmlText
XmlAttributeValue.Namespaces.Add ("ns1", "p")
InfoItem.Value = XmlAttributeValue
```

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to an ISmIxfSchema.

ISmIxfSchema:

Creating a Schema with Classes and Class Attributes

The following procedure creates a basic schema file containing classes and class attributes.

1. Get the schema property (see ISmIxfSchema.)
2. Create a class

```
Dim IxfClass as ISmIxfClass
```

```
'Create a Class "documentMaster":  
Set IxfClass = Schema.Classes.Add("DocumentMaster")  
IxfClass.ParentClass = Null  
IxfClass.IsAbstract = False
```

See [Adding a Class to the IXF Schema](#), for more details.

3. Create a Class attribute

```
Dim IxfAttribute as ISmIxfAttribute
```

```
'Create and add attribute "DocumentName":  
Set IxfAttribute = IxfClass.CurrentClassAttributes.Add("DocumentName")  
'Set properties for the attribute:  
IxfAttribute.TypeDefinition.ValueType = dtString  
IxfAttribute.TypeDefinition.StringType.MaxLength = 50  
IxfAttribute.Required = True  
IxfAttribute.IsNullAllowed = False  
IxfAttribute.IsPrimary = True  
'Create and add attribute "Description":  
Set IxfAttribute = IxfClass.CurrentClassAttributes.Add("Description")  
'Set properties for the attribute:  
IxfAttribute.TypeDefinition.ValueType = dtString  
IxfAttribute.Required = False  
IxfAttribute.IsNullAllowed = True
```

See [Adding an Attribute to a Class](#) for more information.

ISmIxfSchema: Defining a ClassBehavior

In this task, you define a ClassBehavior.

1. Add a Class Behavior definition to the IXF Schema, that is, to ClassesBehaviors

```
Dim IxfClassBehavior as ISmIxfClassBehavior
'Create a Class Behavior "link"
Set IxfClassBehavior = Schema.ClassesBehaviors.Add(
mtEmbedded, "http://www.ixfstd.org/std/ns/core/classBehaviors/links/1.0#link")
```

See [Adding a New ClassBehavior](#) for more information.

2. Add attributes to the ClassBehavior definition

```
Dim IxfAttribute as ISmIxfAttribute

'add attribute "object1":
Set IxfAttribute = IxfClassBehavior.Attributes.Add("object1")
'Set properties for the attribute:
IxfAttribute.TypeDefinition.ValueType = dtObjectReference
IxfAttribute.Required = True
IxfAttribute.IsNullable = True

'add attribute "object2":
Set IxfAttribute = IxfClassBehavior.Attributes.Add("object2")
'Set properties for the attribute:
IxfAttribute.TypeDefinition.ValueType = dtObjectReference
IxfAttribute.Required = True
IxfAttribute.IsNullable = True
```

See [Adding an Attribute to a ClassBehavior](#) for more information.

ISmIxfSchema: Declaring usage of a Class Behavior by a class

In this task, you declare a ClassBehavior in a class.

In case the ClassBehavior has been defined separately, you can retrieve the class behavior object by URI and declare it in a class as follows:

```
IxfClassBehavior = Schema.classesBehaviors.ItemByURI(
"http://www.ixfstd.org/std/ns/core/classBehaviors/links/1.0#link")
IxfClass = Schema.classes.Add("myLink")
IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
```

See [Adding a ClassBehavior to a Class](#) for more information.

**ISmIxfSchema:
Defining a DomainBehavior**

See example section.

**ISmIxfSchema:
Adding Standard Behavior to a schema**

See section [ISmIxfSchemaHelper](#) on page [48](#)

**ISmIxfSchema:
Adding miscellaneous information**

To add miscellaneous information ISmIxfInfo to the schema, you use an ISmIxfInfoItem object

```
Dim IxfInfo As ISmIxfInfo
Dim IxfInfoItem As ISmIxfInfoItem

' Get the Info object
Set IxfInfo = Schema.Info
'Create and return an InfoItem with the indicated name and namespace:
Set IxfInfoItem = IxfInfo.GetInfoItem("transaction", "http://www.vendor.org")
IxfInfoItem.ValueType = dtInt
IxfInfoItem.Value = "2352"
```

SmIxfInitializationData

A **SmIxfInitializationData** object represents initialization of data for SmartIXF applications.

Each creatable object has a reference to the interface **ISmIxfInitializationData**.

Setting Proxy Information

For the present release, the **SmIxfInitializationData** interface relates to the initialization of proxy information for downloading files by an IXF Application installed on a UNIX system (optional on Windows).

The user can obtain the current proxy value by calling the **GetProxy** method.

The **SetProxy** method should be performed when the user wants to indicate the proxy that is about to be used.

On Unix platforms the proxy must be set if files are about to be downloaded from the web.

On windows using this interface is optional since windows can automatically detect and identify a proxy.

Methods

The **SmIxfInitializationData** has the methods

Methods	Description
GetProxy	Returns the value of the proxy string.
SetProxy	Sets the specified string as the proxy string.

Example

In order to set a proxy after creating an object, the **SetProxy** method should be called as described in the following sample code.

```
Public Const PROXY_STR = "123.45.678.90:8080" 'A string indicating the proxy to be used when downloading files from the web.
```

```
Dim IxfWriter As SmIxfWriter
Dim IxfReader As SmIxfReader
Dim IxfStdHelper As SmIxfStdHelpert
Dim IxfExternalSchemaWriter As SmIxfExternalSchemaWriter
```

```
Dim IxfExternalSchemaReader As SmIxfExternalSchemaReader
Dim ProxyStr As String

Set IxfWriter = CreateObject("SmartIXF1.SmIxfWriter")
IxfWriter.InitializationData.SetProxy(PROXY_STR)

Set IxfReader = CreateObject("SmartIXF1.SmIxfReader")
IxfReader.InitializationData.SetProxy(PROXY_STR)

Set IxfStdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")
IxfStdHelper.InitializationData.SetProxy(PROXY_STR)

Set IxfExternalSchemaWriter = CreateObject("SmartIXF1.ExternalSchemaWriter")
IxfExternalSchemaWriter.InitializationData.SetProxy(PROXY_STR)

Set IxfExternalSchemaReader = CreateObject("SmartIXF1.ExternalSchemaReader")
IxfExternalSchemaReader.InitializationData.SetProxy(PROXY_STR)

...

ProxyStr = IxfReader.InitializationData.GetProxy()
```

SmIxfWriter

An **SmIxfWriter** object is used for:

- Creating an IXF Archive file

- Creating and writing a Schema Document (schema file)

- Creating and writing an IXF Instance Document (data file)

- Packaging the schema and data files in the IXF Archive file.

Optionally, the SmIxfWriter can refer to an existing schema file.

Object Diagram

The object diagram of SmIxfWriter is shown below:

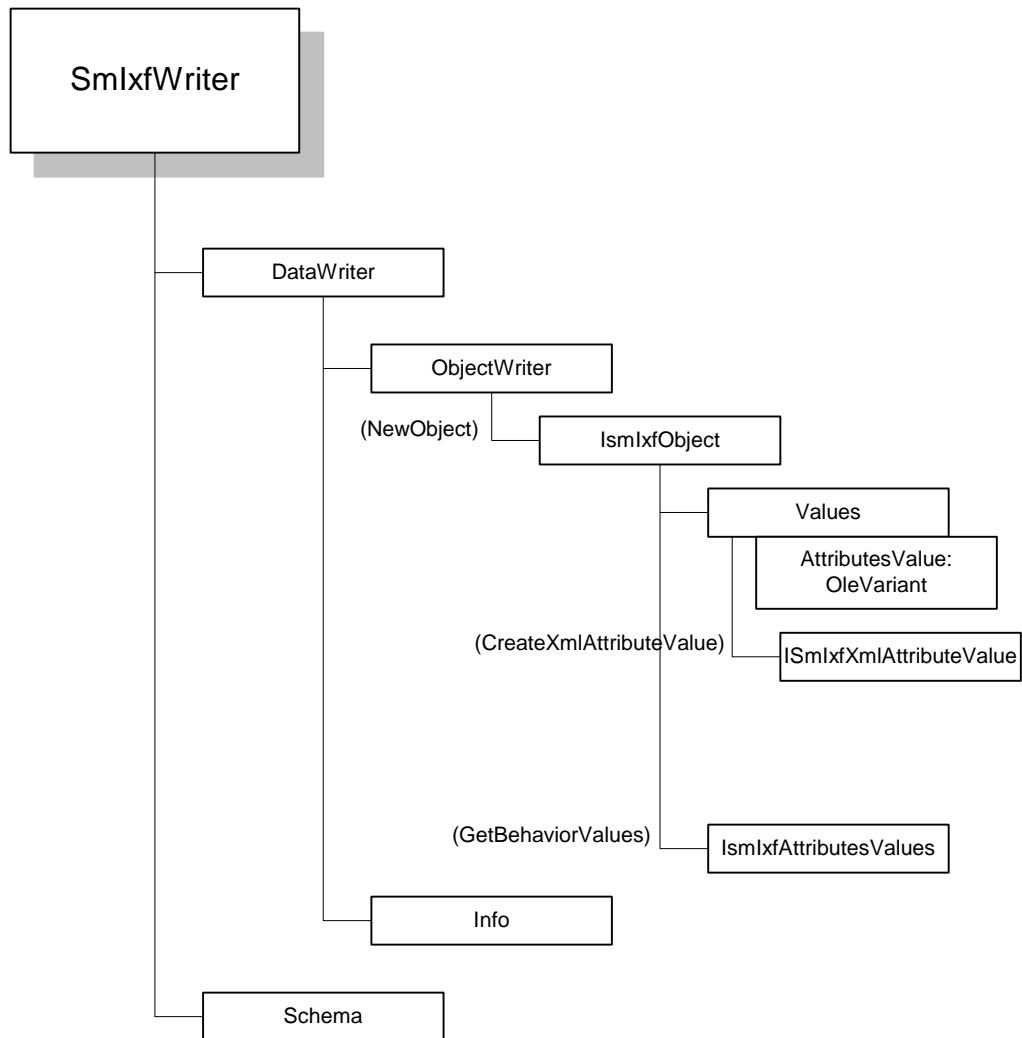


Figure 2-5 SmlxfWriter Object Diagram

Properties

SmIxfWriter has the following properties:

Property	Description
Schema	Holds the definition of the data structure The schema file describes the data model, including its classes and behaviors.
DataWriter	Writes the data file. The data file contains a set of objects that conforms to the data model described in the schema and miscellaneous data. [associated files come from WriterHelper]
InitializationData	Provides access to methods for initializing data for IXF applications. Returns ISmIxfInitializationData

The schema and data files are packaged in an IXF Archive by the SmIxfWriter object.

Methods

The SmIXfWriter has the methods

Methods	Description
CreateIxfArchiveFile	Creates the specified iXF Archive file.
CloseIxfArchiveFile	Closes the iXF Archive file.
SetSchemaMode	Sets the packaging mode of the schema in the Archive file.

Creating the SmIxfWriter Object

To create an SmIxfWriter Object:

```
Dim IxfWriter As SmIxfWriter
Set IxfWriter = CreateObject("SmartIXF1.SmIxfWriter")
```

Creating an iXF Archive File

As described in the iXF Specification, an iXF Archive file is a zip file containing a data file and possibly a schema file. When you use the SmIxfWriter to write an IXF Instance file, you need to create an iXF Archive file to contain the IXF Instance file and possibly the schema file.

Specifying the Schema Packaging State

When you create an iXF Archive file, you first need to specify how the associated schema is to be packaged, using the `SetSchemaMode` method of the `ISmIxfWriter`.

The following table describes the packaging states of the schema file and the software constant used for each state. The packaging state of the schema file is described by the `Mode` parameter of the `SetSchemaMode` method.

Packaging State	Description	SchemaModeEnum Software Constant
Embedded	The schema definition is created and saved to a schema file, which is embedded in the iXF archive file.	mtEmbedded
External	The schema file is not embedded in the iXF package; it was previously defined externally in a file and specified by its SchemaURI.	mtExternal

SetSchemaMode Method

Use the `SetSchemaMode` method to specify the packaging state of the schema. It is called as follows:

```
IxfWriter.SetSchemaMode(mode,[SchemaURI],[SchemaLocation = ""] [Load=True])
```

The arguments of the method are:

Argument	Description
Mode	Packaging state of the schema, one of <code>SchemaModeEnum</code> . See table above.
SchemaURI	The schema namespace. Specified only when <code>Mode</code> is <code>mtExternal</code> .
SchemaLocation	The schema physical location. Specified only when <code>Mode</code> is <code>mtExternal</code> .
Load	Whether or not the schema needs to be loaded. The default is false. Can be set <code>True</code> only when <code>Mode</code> is <code>mtExternal</code> .

Note: If the `Mode` is `mtEmbedded`, or if `Mode` is `mtExternal` and `Load = False` (for example, when the schema is not accessible) then the `Schema` object in the `ixfWriter` object should be populated by hand. See `ISmIxfSchema`.

Creating the IXF Archive

To create the iXF Archive, after calling the SetSchemaMode method, use the methods IxfWriter.CreateIxfArchiveFile and IxfWriter.CloseIxfArchiveFile.

For an example of how to create the IXF Archive, see Common Tasks, SmIxfWriter: Creating an iXF Archive.

ISmIxfDataWriter

The ISmIxfDataWriter Object writes the object and miscellaneous data corresponding to the schema file.

Properties

The ISmIxfDataWriter Object has the two properties:

Property	Description
ObjectWriter	Returns an ISmIxfObjectWriter object, used to write objects to the data file.
Info	Returns an ISmIxfInfo object. It is used for writing miscellaneous information to the data file.

Obtaining the ISmIxfDataWriter Object

To obtain the ISmIxfDataWriter Object from the IxfWriter Object:

```
Dim DataWriter as ISmIxfDataWriter  
Set DataWriter = IxfWriter.DataWriter
```

ISmIxfObjectWriter

The ISmIxfObjectWriter uses the NewObject method to create objects, which are instantiations of the classes declared in the schema.

Obtaining the ISmIxfObjectWriter Object

To obtain the ISmIxfObjectWriter object from the ISmIxfDataWriter object:

```
Dim ObjectWriter as ISmIxfObjectWriter
Set ObjectWriter = IxfWriter.DataWriter.ObjectWriter
```

Creating a New Object

Use the NewObject method to create an object, which is an instantiation of a class defined in the Schema. Use the Class Name and provide a unique Object Id (see next section for more details about the parameters).

```
Set IxfObject = ObjectWriter.NewObject(ixfClassName, ObjectId)
```

ISmIxfObject

The ISmIxfObject object represents an instantiation of a class in the schema. You create it an ISmIxfObject object by specifying the class from which the object is to be instantiated and providing an object id (see previous section).

Use the ISmIxfObject object to access class attributes and class behavior attributes that were declared in the schema file for the object's class.

Properties

The ISmIxfObject has the properties:

Property	Description
Id	Input string that uniquely identifies the object within the IXF Instance Document. Must be a valid NCName. The Object ID value must follow the rules defined for the ID Datatype in XML Schema Part 2: Datatypes, Section 3.3.8: ID.
IxfClassName	Name of class of which this object is an instantiation.
Values	Returns object ISmIxfAttributesValues, which is the set of values of the class attributes for the object's class.

Methods

The ISmIxfObject has the methods:

Method	Description
GetBehaviorValues	Returns object ISmIxfAttributesValues, which is the set of the ClassBehavior attribute values for Class Behaviors declared by the object's class.
Save	Saves object to data file. Can be used only during iXF generation (writing)

ISmIxfAttributesValues

The ISmIxfAttributesValues object, which is returned by the Values and GetBehaviorValues methods of an IxfObject, represents the collection of values of class attributes or ClassBehavior attributes of the IxfObject. You refer to an individual item of the ISmIxfAttributesValues by the name of the corresponding ISmIxfAttribute, which was assigned in the class or ClassBehavior definition in the schema.

```
IxfObject.Values.Item(AttributeName) = some variant value
Set BehaviorValues = IxfObject.GetBehaviorValues(BehaviorURI)
BehaviorValues.Item(AttributeName) = some variant value
```

For an example of how to assign values to class attributes, see Common Tasks, ISmIxfDataWriter: Creating a Data File with Objects and Info.

ISmIxfXmlAttributeValue Object

If you defined the TypeDefinition.ValueType of a class attribute or ClassBehavior attribute as dtXml in the schema definition, you can create an ISmIxfXmlAttributeValue object and assign it as the class attribute value.

```
Set XmlAttributeValue = IxfObject.Values.CreateXmlAttributeValue
```

Note: The ISmIxfXmlAttributeValue object can also be used when using the ISmIxfInfo object to provide an Xml text value to an InfoItem of type dtXml.

Example

```
'In definition of ClassAttributes in Schema, define an Xml attribute:
Set IxfAttribute = IxfClass.CurrentClassAttributes.Add("XmlText")
'Set properties for the attribute:
IxfAttribute.TypeDefinition.ValueType = dtXml
---
'When loading values into Class attributes in Writer:
Dim DataWriter As ISmIxfDataWriter
Dim ObjectWriter As ISmIxfObjectWriter
Dim IxfObject As ISmIxfObject
Dim XmlAttributeValue As ISmIxfXmlAttributeValue
Dim XmlText As String

Set IxfObject = DataWriter.ObjectWriter.NewObject

'Create XmlAttributeValue object and give it an Xml value
XmlAttributeValue = Object.Values.CreateXmlAttributeValue
XmlText = "<p:name>John Bryce</p:name>"
XmlAttributeValue.XML = XmlText
XmlAttributeValue.Namespaces.Add ("ns1", "p")
'Put the XmlAttributeValue object into the class attribute value
IxfObject.Values.Item("XmlText") = XmlAttributeValue
Object.Save
```

ISmIxfInfo

The ISmIxfInfo object represents miscellaneous information written to the data file. See the ISmIxfInfo object of the Schema object.

The ISmIxfInfo object holds miscellaneous information, which cannot be categorized as classes or behaviors. It is a collection of ISmIxfInfoItem objects.

Methods

It has the methods:

Method	Description
GetInfoItem	Gets an InfoItem from the collection by Name and Namespace.
Save	Saves the InfoItems collection to the iXF Instance file. Can be used only when Info is obtained through IxfWriter.DataWriter, i.e., during iXF generation.
CreateXmlAttributeValue	Creates an ISmIxfXmlAttributeValue object

Note: This ISmIxfInfo object under DataWriter represents an independent way to write miscellaneous data, which you can use instead of or in addition to the ISmIxfInfo object under the ISmIxfSchema object. The difference is that with this ISmIxfInfo object you need to save the object, as shown below.

For an example of how to write an Info object, see [Common Tasks, ISmIxfDataWriter: Writing an Info section](#).

Note: The ISmIxfInfo information must be written to the data file prior to any object information.

ISmIxfSchema

The ISmIxfSchema object represents the schema file in the IXF Archive being written. See [ISmIxfSchema](#) on page 2.

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to a SmIxfWriter.

SmIxfWriter: Creating an iXF Archive

As described in the iXF Specification, an iXF Archive file is a zip file containing a data file and possibly a schema file.

To create an iXF package file:

1. Create the IxfWriter object

```
Dim IxfWriter As SmIxfWriter
Set IxfWriter = CreateObject("SmartIXF1.SmIxfWriter")
```

2. Set the schema packaging state using the method SetSchemaMode:

Examples of setting the schema packaging mode:

```
IxfWriter.SetSchemaMode mtEmbedded
```

or:

```
IxfWriter.SetSchemaMode mtExternal,
"http://www.vendor.org.schema",
"c:\Schemas\MySchema.xsd", true
```

or :

```
IxfWriter.SetSchemaMode mtExternal,
"http://www.vendor.org.schema",
"c:\Temp\MySchema.xsd", False
```

See [Creating an iXF Archive File](#) for more information.

3. Create a schema (see [ISmIxfSchema](#))

If the ModeTypeEnum is embedded, or if ModeTypeEnum is external and Load = False (for example, when the schema is not accessible) then the Schema object in the ixfWriter object should be populated by hand.

4. Use the method `CreateIxfArchiveFile` to initialize the process of creating an IXF Archive:

```
IxfWriter.CreateIxfArchiveFile "test.ixf"
```

NOTE: This method can be called only after `SetSchemaMode` is called and the schema object is populated.

5. Insert the data information -- Info, objects, changes and files (see next section [ISmIxfDataWriter:Creating a Data File with Objects and Info](#))

6. Close the iXF file:

```
IxfWriter.CloseIxfArchiveFile
```

ISmIxfDataWriter:Creating a Data File with Objects and Info

The following procedure creates a basic data file containing objects and miscellaneous information. It is assumed that a schema has already been created.

The data file is created automatically as part of the package. It is named `IXF_Data.xml`.

In this section you create an object corresponding to a class in the schema and assign values to the class attributes and `ClassBehavior` attributes for class behaviors declared by the class.

Note: If you are writing miscellaneous (Info) information, it must be written first, before any object information.

1. Create an object

To create an object, you need to obtain the `ISmIxfObjectWriter` Object as follows:

```
Dim ObjectWriter As ISmIxfObjectWriter
```

```
Set ObjectWriter = IxfWriter.DataWriter.ObjectWriter
```

Use the `NewObject` method of the `ISmIxfObjectWriter` Object to create the new object, where you specify the `ClassName` and provide a unique `ObjectId`. The `NewObject` method returns an object of type `ISmIxfObject`.

```
' Create an object for the data file
```

```
Dim IxfObject as ISmIxfObject
```

```
Set IxfObject = ObjectWriter.NewObject("DocumentMaster", "OID_1")
```

2. Assign values according to the object's class attributes

The attribute values are stored in the collection object `ISmIxfAttributesValues`. This object is obtained from `ISmIxfObject` as follows:

```
AttributesValues = ixfObject.Values
```

You assign a value to this object.

```
' Assign values to the object's attributes
AttributesValues.Item("DocumentName") = "MyDocument"
```

3. Assign values to the `ClassBehavior` attributes

To assign values to the `ClassBehavior` attributes, use the method `GetBehaviorValues` of `ISmIxfObject`. The method returns the object `ISmIxfAttributesValues` as in the previous step.

```
Dim BehaviorValues As ISmIxfAttributesValues
```

```
IxfLinkObject = IxfWriter.DataWriter.ObjectWriter.NewObject("MyLink", "OID_2")
Set BehaviorValues =
IxfLinkObject.GetBehaviorValues(http://project/behavior1#link)
```

You assign values to this object as in the previous step.

```
BehaviorValues.Item("object1") = IxfObject1
```

4. Save the object.

```
ixfObject.Save
```

Note: Save each object as soon as you are finished creating it.

SmlxfWriter: Creating a ISmlxfXmlAttributeValue

```
Dim XmlAttributeValue as ISmlxfXmlAttributeValue
```

```
Set XmlAttributeValue = IxfWriter.DataWriter.Info.CreateXmlAttributeValue  
XmlAttributeValue.XML = <p:name>John Bryce<p:name>  
XmlAttributeValue.Namespaces.Add 'http://www.vendor.com/ns/personalIdentity', 'p'
```

ISmlxfDataWriter: Writing an Info section

'Optional "Info" section:

```
Set InfoItem = Writer.DataWriter.Info.GetInfoItem("From",  
"http://smarteam.com/dev/ixf/test")  
InfoItem.ValueType = dtString  
InfoItem.Value = "Ann Barkley"  
Set InfoItem = Writer.DataWriter.Info.GetInfoItem("To",  
"http://smarteam.com/dev/ixf/test")  
InfoItem.ValueType = dtString  
InfoItem.Value = "Bruce Mayer"  
Set InfoItem = Writer.DataWriter.Info.GetInfoItem("Subject",  
"http://smarteam.com/dev/ixf/test")  
InfoItem.ValueType = dtString  
InfoItem.Value = "iXF Example"  
DataWriter.Info.Save
```

Note: If you are writing Info to the data file, it must be saved to the data file before saving any object to the data file.

SmlxfReader

A **SmlxfReader** object is used for:

- Unpacking an IXF Archive file

- Reading a Schema Document (schema file)

- Reading an IXF Instance Document (data file)

Optionally, the SmlxfReader can refer to an external schema file.

Object Diagram

The object diagram of SmlxfReader is shown below:

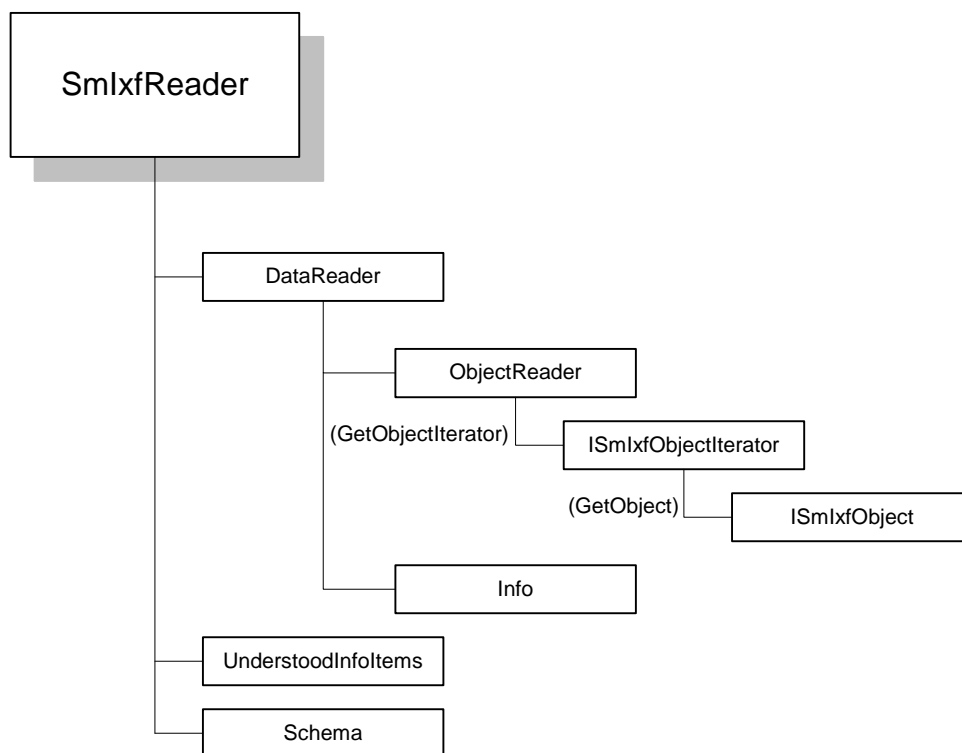


Figure 2-6 SmlxfReader Object Diagram

Properties

The SmIXfReader has the properties

Property	Description
DataReader	Reference to ISmIXfDataReader, which reads the data file. The data file contains a set of objects and miscellaneous data that conforms to the data model described in the schema. [Associated files come from Reader helper].
UnderstoodInfoItems	Collection of InfoItems that the DataReader declares as understood. Used to validate read-in InfoItems marked as “mustUnderstand”.
Schema	Reference to ISmIXfSchema, which holds the definition of the data structure.
ValidateMustUnderstand	If true, validate read-in InfoItems marked as “mustUnderstand” against the UnderstoodInfoItems collection.
InitializationData	Provides access to methods for initializing data for IXF applications. Returns ISmIXfInitializationData

Methods

The SmIXfReader has the methods

Methods	Description
OpenIxfArchiveFile	Opens the specified iXF Archive file for reading.
Close	Closes the iXF Archive file for reading.

ISmIXfDataReader

The ISmIXfDataReader object reads the object and miscellaneous data from the data file corresponding to the schema file. The ISmIXfDataReader object includes the ObjectReader property, which is used to read objects from the data file by iteration, using the ObjectsIterator property.

Properties

The ISmIxfDataReader Object has the two properties:

Property	Description
ObjectReader	Returns an ISmIxfObjectReader object
Info	Miscellaneous (Info) information read from the data file.

Obtaining the ISmIxfDataReader Object

To obtain the ISmIxfDataReader Object from the ixfReader Object:

```
Dim DataReader As ISmIxfDataReader
Set DataReader = IxfReader.DataReader
```

ISmIxfObjectReader

The ISmIxfObjectReader object reads objects from the data file. It has one method GetObjectIterator, which returns the object ISmIxfObjectIterator.

Obtaining the ISmIxfObjectReader Object

To obtain the ISmIxfObjectReader Object:

```
Dim IxfObjectReader as ISmIxfReader
Set IxfObjectReader = DataReader.ObjectReader
```

ISmIxfObjectIterator

The ISmIxfObjectIterator reads the objects one-by-one from the data file.

Use the GetObjectIterator method to get an ISmIxfObjectIterator from the ObjectReader as follows:

```
Set ObjectIterator = IxfReader.DataReader.ObjectReader.GetObjectIterator
```

Properties and Methods

The ISmIxfObjectIterator has one property and three functions:

Property	Description
AtEnd	Indicates whether the iterator has reached the end of the collection.
Method	Description
GetObject	Returns the object to which the iterator is currently pointing.
Next	Sets the iterator to read the next object in the collection

ISmIxfObject

The object represents an individual object read from the data file. See [ISmIxfObject](#) under ISmIxfObjectWriter.

Use the GetObject method to get an ISmIxfObject from the ObjectIterator as follows:

```
IxfObject = ObjectIterator.GetObject
```

For an example of how to use the ObjectIterator to read objects, see Common Tasks, SmIxfReader: Reading an iXF Package.

ISmIxfInfo

The ISmIxfInfo object represents miscellaneous information read from the data file. See [ISmIxfInfo](#) on page [19](#).

Note: The ISmIxfInfo information in the IXF Instance file, if it exists, must be read before all object information.

For an example of how to read ISmIxfInfo objects, see Common Tasks, SmIxfReader: Reading an iXF Package.

ISmIxfUnderstoodInfoItems

ISmIxfUnderstoodInfoItems is a collection object, prepared by the user of the ISmIxfReader object, of items of type ISmIxfUnderstoodInfoItem, which denote InfoItems that are required to be understood.

The ISmIxfUnderstoodInfoItems corresponds to a list that specifies those InfoItems that he declares he understands. When the InfoItems are read by the ISmIxfReader, the MustUnderstand property of each InfoItem is matched with the corresponding InfoItem entry in the ISmIxfUnderstoodInfoItems list. If the MustUnderstand property of an InfoItem is true and the corresponding InfoItem entry is not found in ISmIxfUnderstoodInfoItems, the reading process is stopped.

Properties

The ISmIxfUnderstoodInfoItem Object has the two properties:

Property	Description
Name	Name of the understood item.
Namespace	Namespace of the understood item.

ISmIxfSchema

The SmIxfSchema object represents the schema file in the package being read. See [ISmIxfSchema](#) on page [2](#).

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to a SmIxfReader.

SmIxfReader: Reading an iXF Package

As described in the IXF Specification, an iXF Archive file is a zip file containing a data file and possibly a schema file. In order to read an iXF archive file proceed as follows.

1. Create an IxfReader Object

```
Dim IxfReader as ISmIxfReader
Set IxfReader = CreateObject("SmartIxf1.SmIxfReader")
```

2. Open an Ixf archive file:

```
IxfReader.OpenIxfArchiveFile "test.ixf", True
```

3. Read Info (if it exists)

```
Dim Info as ISmIxfInfo
Dim SenderName, ReceiverName, Subject as Variant

Set Info = IxfReader.DataReader.Info
Set InfoItem = Info.GetInfoItem("From", "http://smarteam.com/dev/ixf/test")
SenderName = InfoItem.Value
Set InfoItem = Info.GetInfoItem("To", "http://smarteam.com/dev/ixf/test")
ReceiverName = InfoItem.Value
Set InfoItem = Info.GetInfoItem("Subject", "http://smarteam.com/dev/ixf/test")
Subject = InfoItem.Value
```

4. Read Objects:

```
Dim ObjectIterator as ISmIxfObjectIterator
Dim IxfObject as ISmIxfObject
Set ObjectIterator = IxfReader.DataReader.ObjectReader.GetObjectIterator
While ObjectIterator.AtEnd = False
    Set IxfObject = ObjectIterator.GetObject
    ....
    ObjectIterator.Next
Wend
```

5. Close the reader object:

```
IxfReader.Close
```

Reading and Writing an External Schema

The SmartIxf library provides two objects for reading and writing an external schema.

SmIxfExternalSchemaWriter

The SmIxfExternalSchemaWriter has the three properties:

Property	Description
Schema	Returns object ISmIxfSchema containing the external schema information.
SchemaURI	URI of external schema file.
Initialization Data	Provides access to methods for initializing data for IXF applications. Returns ISmIxfInitializationData

The SmIxfExternalSchemaWriter has one method, Save(FileName), which saves the schema to the file FileName.

See [ISmIxfSchema](#) on page 2, for more information.

SmIxfExternalSchemaReader

The SmIxfExternalSchemaReader handles reading an external iXF schema document

The SmIxfExternalSchemaReader has one method:

Load(SchemaLocation), which loads the external schema with the specified SchemaLocation into the ISmIxfSchema object.

and one property:

InitializationData, which provides access to methods for initializing data for IXF applications. Returns ISmIxfInitializationData

See [ISmIxfSchema](#) on page 2, for more information.

ISmIxfStdHelper

IXF Standard Behaviors, as defined in the IXF Specification, Section 4, are a set of Class Behaviors and Domain Behaviors, which provide common functionality required by many IXF-enabled applications.

The ISmIxfStdHelper object provides methods to simplify and facilitate the usage of IXF Standard Behaviors in the following main functional areas:

ISmIxfSchemaHelper – Adding Standard Behavior definitions to a schema

ISmIxfWriterHelper – Using Standard Behaviors while writing IXF documents

ISmIxfReaderHelper – Using Standard Behaviors while reading IXF documents

Methods

The ISmIxfStdHelper object provides the following methods:

Method	Description
CreateReaderHelper	Creates a reader Helper for using Standard Behaviors while reading IXF documents:
CreateSchemaHelper	Creates a schema Helper for adding Standard Behavior definitions to a schema.
CreateWriterHelper	Creates a writer Helper for using Standard Behaviors while writing IXF documents.xxx
InitializationData	Provides access to methods for initializing data for IXF applications. Returns ISmIxfInitializationData.

Obtaining the SmlxfStdHelper Object

```
Dim StdHelper as ISmIxfStdHelper  
StdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")
```

Standard Behaviors

The SmartIxf Library supports the following mechanisms, which are defined in the iXF standard:

Time Stamping – provides the ability to time-stamp IXF Objects.

Change Tracking – provides a standard mechanism for tracking changes in an IXF Instance Document

File Association – provides a standard mechanism for:

- Storing file information in the IXF Instance file
- Embedding files in an IXF Archive file
- Associating IXF Objects with files.

Versioning – provides the ability to tag iXF Objects with versioning information.

Links – to formalize and classify the relationships between objects in an IXF Instance Document.

ISmIxfSchemaHelper

The ISmIxfSchemaHelper object provides methods to support defining Standard Behaviors in a schema.

The ISmIxfSchemaHelper object uses the following naming convention to describe its methods:

Method	Description
Add[<i>std-behavior-name</i>]Support	Adds the classes, class behaviors, and domain behaviors required to support the Standard Behavior to the schema. Note that all implementation details regarding the Classes, which are added by this method to the schema, may change in subsequent versions of this API.
Enable[<i>std-behavior-name</i>]ForClass	Enables standard behavior functionality in a specific user-defined class.
Is[<i>std-behavior-name</i>]Enabled	Tests a user-defined class to see if it is associated with a Standard Behavior.

Change-Tracking Standard Behavior

The SmartIxf Library provides an implementation of the Change-Tracking Standard Behavior to provide a standard mechanism for tracking changes on

objects in an IXF Instance Document, including object creation, object deletion, and attribute value modification.

Methods

The ISmIxfSchemaHelper object provides the following methods to support defining the Change Tracking Standard Behavior in a schema:

Method	Description
AddDefaultChangesSupport	Adds the Change-Tracking Standard Behavior to the schema as shown in Change-Tracking Standard Behavior .
EnableChangeTrackingForClass	Add the TrackChanges Class Behavior to the specified user-defined class, enabling the class to be change-tracked.
IsChangeTrackingEnabled	Returns true if the specified user-defined class supports the Change Tracking Standard Behavior.

Behaviors

The following table shows the domain and class behaviors added to the schema that support the implementation of the Change Tracking Standard Behavior:

Domain Behavior	
Name	URI
Change Tracking	<ixfstdns>/domainBehaviors/changeTracking/1.0

Class Behaviors	
Name	URI
change	<code><ixfstdns-c>/changeTracking/1.0#change</code>
objectDeleted	<code><ixfstdns-c>/changeTracking/1.0#objectDeleted</code>
objectValue Modified	<code><ixfstdns-c>/changeTracking/1.0#objectValueModified</code>
objectCreated	<code><ixfstdns-c>/changeTracking/1.0#objectCreated</code>
transaction	<code><ixfstdns-c>/changeTracking/1.0#transaction</code>

Note: An object can be change-tracked only if it instantiates a class that is enabled for change-tracking.

File Association Standard Behavior

The SmartIxf Library provides an implementation of the File Association Standard Behavior to provide a standard mechanism for:

Storing file information in an IXF Instance file, including

- File Name
 - Physical Location
 - MIME Content Type
- Associating an IXF Object with a specific file
- Distinguishing between main and secondary files

Embedding files in an IXF Archive file

Methods

The ISmIxfSchemaHelper object provides the following methods to support defining the File Association Standard Behavior in a schema:

Method	Description
AddDefaultFilesSupport	Adds the Files Standard Behavior to the schema as shown in File Association Standard Behavior, including the Class Behaviors:
EnableFileAssociationForClass	Add the File Association Class Behavior to the specified user-defined class.
IsFileAssociationEnabled	Returns true if the specified user-defined class supports the Files Standard Behavior.

Behaviors

The following table shows the domain and class behaviors added to the schema that support the implementation of the File Association Standard Behavior:

Domain Behavior	
Name	URI
Files	<ixfstdns-d>/domainBehaviors/files/1.0

Class Behaviors	
Name	URI
File Association	<ixfstdns-c>/files/1.0#fileAssociation
File Description	<ixfstdns-c>/files/1.0#fileDescription
Main File	ixfstdns-c>/files/1.0#mainFile
Secondary File	<ixfstdns-c>/files/1.0#secondaryFile
Transaction	<ixfstdns-c>/changeTracking/1.0#transaction

Note: An object can be associated with a file only if it instantiates a class that is enabled for File Association.

Versioning Standard Behavior

The SmartIxf Library provides an implementation of the Versioning Standard Behavior to provide the ability to tag IXF Objects with versioning information, enabling you to identify successive revisions of the same master entity.

The versioning information for an object includes the version identifier of the current version of the object and the version identifier of its previous version.

The Versioning Standard Behavior is different from the Change-Tracking Standard Behavior: it just assigns version numbers without tracking the changes between versions.

Methods

The ISmIxfSchemaHelper object provides the following methods to support defining the Versioning Standard Behavior in a schema:

Method	Description
AddDefaultVersioningSupport	Adds the Versioning Standard Behavior to the schema.
EnableVersioningForClass	Add the Versioning Class Behavior to the specified user-defined class.
IsVersioningEnabled	Returns true if the specified user-defined class supports the Versioning Standard Behavior.

Behaviors

The following table shows the class behaviors added to the schema that support the Versioning Standard Behavior:

Class Behaviors	
Name	URI
Versioning	<ixfstdns-c>/versioning/1.0#version

Note: An object can be versioned only if it instantiates a class that is enabled for Versioning.

TimeStamp Standard Behavior

The SmartIxf Library provides an implementation of the TimeStamp Standard Behavior to provide the ability to tag IXF Objects with TimeStamp information, enabling you to mark the time of object creation.

Methods

The ISmIxfSchemaHelper object provides the following methods to support defining the TimeStamp Standard Behavior in a schema:

Method	Description
AddDefaultTimeStampSupport	Adds the TimeStamp Standard Behavior to the schema, as shown in TimeStamp Standard Behavior.
EnableTimeStampForClass	Add the enabler TimeStamp Class Behavior to the specified user class.
IsTimeStampEnabled	Returns true if the specified user class supports the TimeStamp Standard Behavior.

Behaviors

The following table shows the class behaviors added to the schema that support the TimeStamp Standard Behavior:

Class Behaviors	
Name	URI
Time Stamping	<ixfstdns-c> /timeStamp/1.0#timeStamp

Note: An object can be time stamped only if it instantiates a class that is enabled for TimeStamp.

Obtaining the ISmIxfSchemaHelper Object

To create an ISmIxfSchemaHelper Object:

```
Dim SchemaHelper as ISmIxfSchemaHelper
'Create Schema Helper:
Set SchemaHelper = StdHelper.CreateSchemaHelper(Schema)
```

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to a `ISmIxfSchemaHelper`.

ISmIxfSchemaHelper:

Add supported standard behaviors to the schema

1. Write basic schema, see Common Tasks in [ISmIxfSchema](#) section.

2. Get SchemaHelper object

```
Dim StdHelper as ISmIxfStdHelper
Dim SchemaHelper as ISmIxfSchemaHelper
```

```
'Create stdHelper:
StdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")
```

```
'Create Schema Helper:
Set SchemaHelper = StdHelper.CreateSchemaHelper(Schema)
```

3. Add support for standard behaviors to the schema:

```
'Add support for Standard Behaviors:
SchemaHelper.AddDefaultChangesSupport
SchemaHelper.AddDefaultFilesSupport
SchemaHelper.AddDefaultVersioningSupport
SchemaHelper.AddDefaultTimeStampSupport
```

4. Enable a user-defined class to support standard behaviors

```
SchemaHelper.EnableFileAssociationForClass (IxfClass)
SchemaHelper.EnableChangeTrackingForClass (IxfClass)
SchemaHelper.EnableTimeStampForClass (IxfClass)
```

ISmIxfWriterHelper

The ISmIxfWriterHelper object supports writing Standard Behaviors attribute information when writing a data file.

Object Diagram

The object diagram of ISmIxfWriterHelper is shown below:

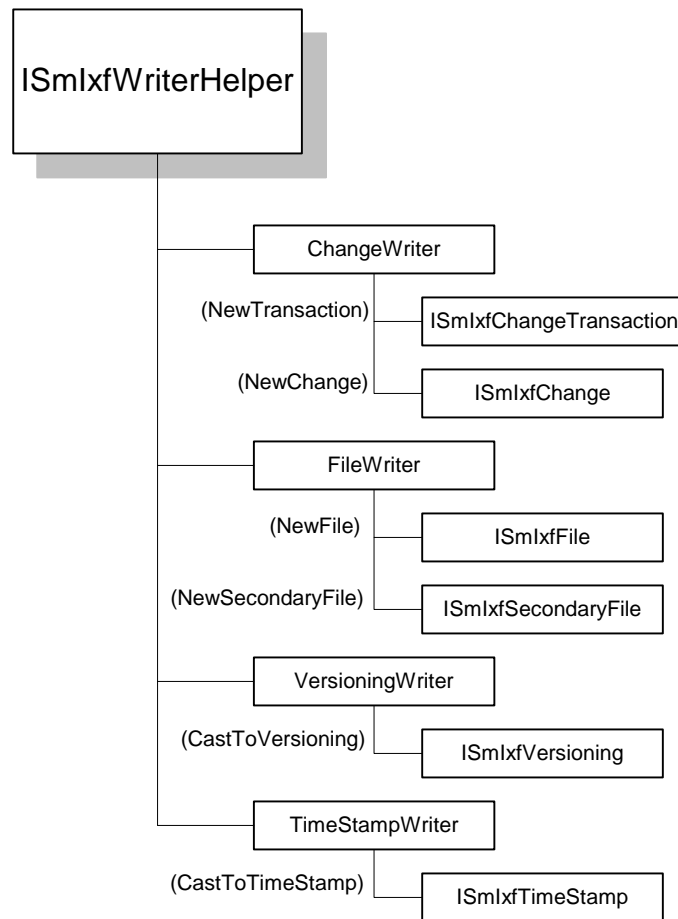


Figure 2-7 ISmIxfWriterHelper Object Diagram

Properties

Four WriterHelper properties are provided corresponding to the supported Standard Behaviors:

Property	Description
ChangeWriter	Provides methods for writing Change-Tracking information
FileWriter	Provides methods for writing File Association information
VersioningWriter	Provides methods for writing Versioning information
TimeStampWriter	Provides methods for writing TimeStamp information

Obtaining the ISmIxfWriterHelper Object

To create an ISmIxfWriterHelper Object:

```
Dim WriterHelper as ISmIxfWriterHelper
'Create Writer Helper:
Set WriterHelper = StdHelper.CreateWriterHelper(IxfWriter)
```

ISmIxfChangeWriter

The ISmIxfChangeWriter writes change-tracking information to the data file.

Note: In order to use the methods of the ISmIxfChangeWriter you need to have added the ChangeTracking Standard Behavior support in the schema, using the SchemaHelper method AddDefaultChangesSupport. In addition, in order to use Change-Tracking on a specific IxfObject, you need to have enabled the ChangeTracking Standard Behavior support for the class that this object instantiates, using the SchemaHelper method EnableChangeTrackingForClass (see Change-Tracking Standard Behavior)

Obtaining the ISmIxfChangeWriter Object

```
Dim ChangeWriter as ISmIxfChangeWriter
Set ChangeWriter = WriterHelper.ChangeWriter
```

Methods

The ISmIxfChangeWriter object provides the following methods to write Change-Tracking information to the data file:

Method	Description
NewChange	Creates an ISmIxfChange object
NewTransaction	Creates an ISmIxfChangeTransaction

ISmIxfChangeTransaction

The ISmIxfChangeTransaction Object is a container object that represents a group of changes, where each change is represented by a ISmIxfChange object. An ISmIxfChange object is linked to an ISmIxfChangeTransaction Object by its Transaction Id property.

Obtaining a ISmIxfChangeTransaction Object

To create a ISmIxfChangeTransaction Object:

```
Dim Transaction as ISmIxfChangeTransaction  
Set Transaction = ChangeWriter.NewTransaction(Id)
```

Note: ISmIxfChangeTransaction objects can also be obtained through ISmIxfChangeReader Object. See ChangeReader.

Save a ChangeTransaction to the data file as follows:

```
ChangeTransaction.Save
```

Properties

The ISmIxfChangeTransaction Object has the properties

Property	Description
Id	Object Id of the SmIxfChangeTransaction object. Has to be a valid NCName.
ParentTransactionId	Object Id of parent SmIxfChangeTransaction object in the file, if it exists
PreviousTransactionId	Object Id of the previous SmIxfChangeTransaction object in the file, if it exists.

Methods

The ISmIxfChangeTransaction Object has the methods

Property	Description
Save	Saves the ChangeTransaction to the data file.

ISmIxfChange

The ISmIxfChange object represents an individual change on an object, and includes change-tracking information. Each ISmIxfChange object is associated with some ISmIxfChangeTransaction object by the Transaction Id property. Therefore, an ISmIxfChangeTransaction object needs to be created first.

Three types of changes are tracked: object creation, object deletion and object modification. The actual changes for each type are represented by three separate objects, which are returned as properties of ISmIxfChange:

- ISmIxfObjectCreated – which contains the Object Id of the created object
- ISmIxfObjectDeleted – which contains a reference to the deleted object
- ISmIxfObjectValueModified – This contains the Id of the modified object and contains a list of the object attributes that were changed, including their previous values.

Object Diagram

The object diagram of ISmIxfChange is shown below:

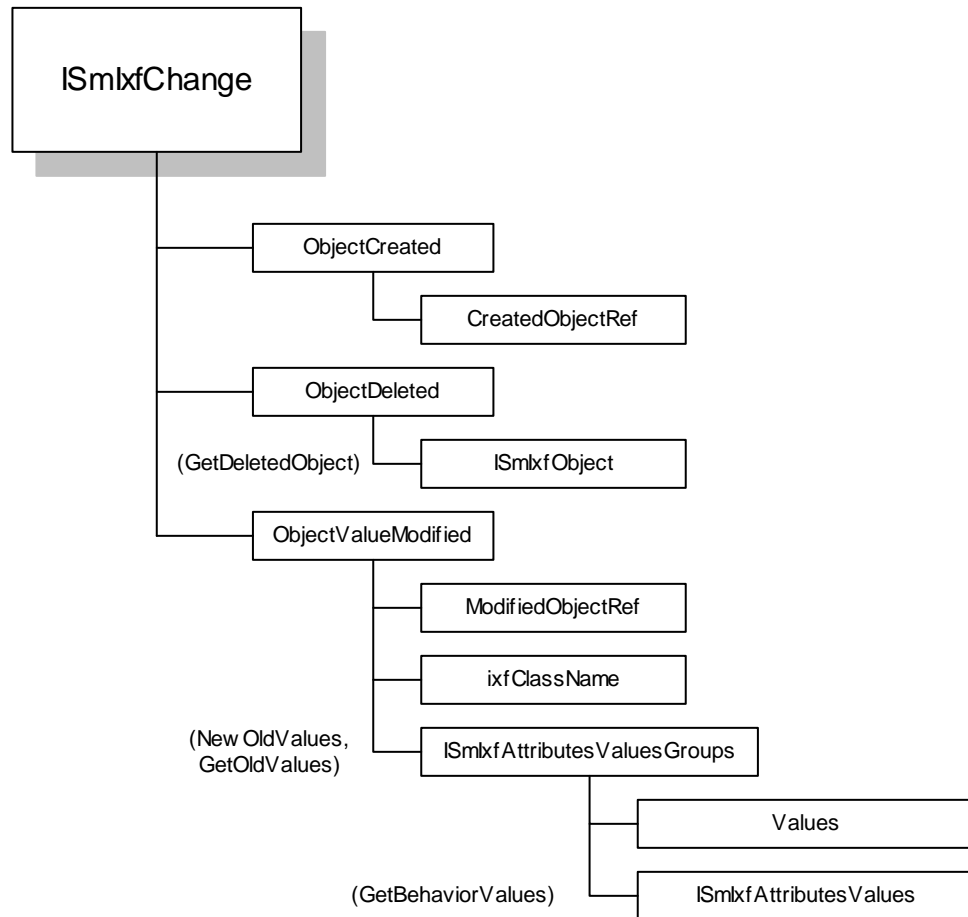


Figure 2-8 ISmIxfChange Object Diagram

Obtaining a ISmIxfChange Object

To create a ISmIxfChange Object:

```
Dim Change as ISmIxfChange
Set Change = ChangeWriter.NewChange(Id, ChangeType, TransactionId)
```

Note: ISmIxfChange objects can also be obtained through ISmIxfChangeReader Object. See [ChangeReader](#).

Properties

The ISmIxfChange object has the properties:

Property	Description
Id	Object Id of ISmIxfChange object. Has to be a valid NCName.
TransactionId	Object Id of ISmIxfChangeTransaction to which this change belongs
ChangeType	The type of the SmIxfChange, one of ChangeTypeEnum, with the following possible values: <ul style="list-style-type: none">- ctObjectCreated- ctObjectDeleted- ctObjectValueModified
ObjectCreated	Returns ISmIxfObjectCreated object (see below.) Can be accessed only when the Change is of type ctObjectCreated.
ObjectDeleted	Returns ISmIxfObjectDeleted object (see below.) Can be accessed only when the Change is of type ctObjectDeleted.
ObjectValueModified	Returns ISmIxfObjectValueModified object (see below.) Can be accessed only when the Change is of type ctObjectValueModified.
PreviousChangeId	The Object Id of the previous SmIxfChange in time.
PreviousChangeIdPer Object	The Object Id of the previous SmIxfChange on the same object.
IxfObject	Pointer to the ISmIxfObject that is wrapped by the current ISmIxfChange object

Methods

The ISmIxfChange Object has the methods

Property	Description
Save	Saves the Change to the data file.

ISmIxfObjectCreated

The ISmIxfObjectCreated represents a change of type ctObjectCreated. This change is meant to point to a new object that was created and added to an already existing set of objects.

Obtaining a ISmIxfObjectCreated Object

ISmIxfObjectCreated object is obtained through ISmIxfChange object of type ctObjectCreated:

```
Dim Change as ISmIxfChange
Dim ObjectCreated as ISmIxfObjectCreated
Set Change = ChangeWriter.NewChange(Id, ctObjectCreated, TransactionId)
Set ObjectCreated = Change.ObjectCreated
```

The ISmIxfObjectCreated object has the properties:

Property	Description
CreatedObjectRef	The Id of the created object

Example

```
ObjectCreated.CreatedObjectRef = "1"
```

ISmIxfObjectDeleted

The ISmIxfObjectDeleted represents a change of type ctObjectDeleted. This change is meant to hold the information of an object that was deleted from the data objects set.

Obtaining a ISmIxfObjectDeleted Object

ISmIxfObjectDeleted object is obtained through ISmIxfChange object of type ctObjectDeleted:

```
Dim Change as ISmIxfChange
Dim ObjectDeleted as ISmIxfObjectDeleted
Set Change = ChangeWriter.NewChange(Id, ctObjectDeleted, TransactionId)
Set ObjectDeleted = Change.ObjectDeleted
```

Methods

The ISmIxfObjectDeleted object has the methods:

Method	Description
SetDeletedObject	Sets the deleted object as the object referenced by the SmIxfChange
GetDeletedObject	Returns an ISmIxfObject, which is the deleted object.

Example

```
ObjectDeleted.SetDeletedObject (IxfObject)
```

ISmIxfObjectValueModified

The ISmIxfObjectValueModified represents a change of type ctObjectValueModified; it contains the previous values of object attributes that were modified, including both class attributes and behavior attributes.

You do not load individual previous object attribute values directly into the ISmIxfObjectValueModified change object. Instead, you load the previous object attribute values, for the attributes that changed, into the intermediate object ISmIxfAttributesValuesGroups and then map this object to the ISmIxfObjectValueModified object using the method SetOldValues, as described below.

Similarly, when you want to access the previous object attribute values in a ISmIxfObjectValueModified change object, you use the GetOldValues method to extract the information into a ISmIxfAttributesValuesGroups object.

An empty intermediate ISmIxfAttributesValuesGroups object can be created from the ISmIxfObjectValueModified object using the method NewOldValues.

Obtaining a ISmIxfObjectValueModified Object

ISmIxfObjectValueModified object is obtained through ISmIxfChange object of type ctObjectValueModified:

```
Dim Change as ISmIxfChange
Dim ObjectDeleted as ISmIxfObjectValueModified
Set Change = ChangeWriter.NewChange(Id, ctObjectValueModified, TransactionId)
Set ObjectValueModified = Change.ObjectValueModified
```

Properties and Methods

The ISmIxfObjectValueModified object has the following properties and methods:

Property	Description
ModifiedObjectRef	The Object Id of the modified object
IxfClassName	The name of the class that the modified object instantiates
Method	Description
NewOldValues	Creates and returns an ISmIxfAttributesValuesGroups object – an empty collection of attribute values to be filled by the user with the values of the modified object prior to the change (old values)
SetOldValues	Sets the old values of the SmIxfObjectValueModified object to be the values specified in the OldValues argument collection, where OldValues was filled in by the user.
GetOldValues	Returns the collection of attribute values of the modified object prior to the change represented by SmIxfObjectValueModified

ISmIxfAttributesValuesGroups

Collection of ISmIxfAttributesValues objects; where each ISmIxfAttributesValues object is a group of either class attributes or behavior attributes.

Properties and Methods

The ISmIxfAttributesValuesGroups object has the following properties and methods:

Property	Description
Values	A SmIxfAttributesValues object that represents class attributes values (see ISmIxfAttributesValues .)
Method	Description
GetBehaviorValues	A SmIxfAttributesValues object that represents ClassBehavior attributes values (see ISmIxfAttributesValues .)

Example

```
Dim OldValues as ISmIxfAttributesValuesGroups
```

```
ObjectValueModified.ModifiedObjectRef = "1"  
ObjectValueModified.IxfClassName = "DocumentMaster"  
Set OldValues = ObjectValueModified.NewOldValues  
'Inserting an old value for a class attribute that was modified:  
OldValues.Values.Item("DocumentName") = "MyDocument"  
ObjectValueModified.SetOldValues(OldValues)
```

When you finish creating the change, you need to call the Save method of the ISmIxfChange object for all the change details to be saved to the data file:

```
Change.Save
```

For an example of writing a change, see Common Tasks, ISmIxfChangesWriter: Writing a change.

ISmIxfFileWriter

The FileWriter lets you include a file as part of the IXF data. In order to include a file, you must create a ISmIxfFile object to represent it. The ISmIxfFile object contains detailed information about the file, such as its name and location.

The physical file, which is represented by the ISmIxfFile object, can be embedded into the iXF Archive file, using the EmbedFile method.

In addition, you can associate the file with an existing IxfObject such as a Document.

Note: In order to use the methods of the ISmIxfFileWriter you need to have added the File Association Standard Behavior support in the schema. Specifically you should include the AddDefaultFilesSupport method in the schema (see [File Association Standard Behavior](#).) If you want to use the FileAssociation capability you need to include additional methods, as described below.

Object Diagram

The object diagram of ISmIxfFileWriter is shown below:

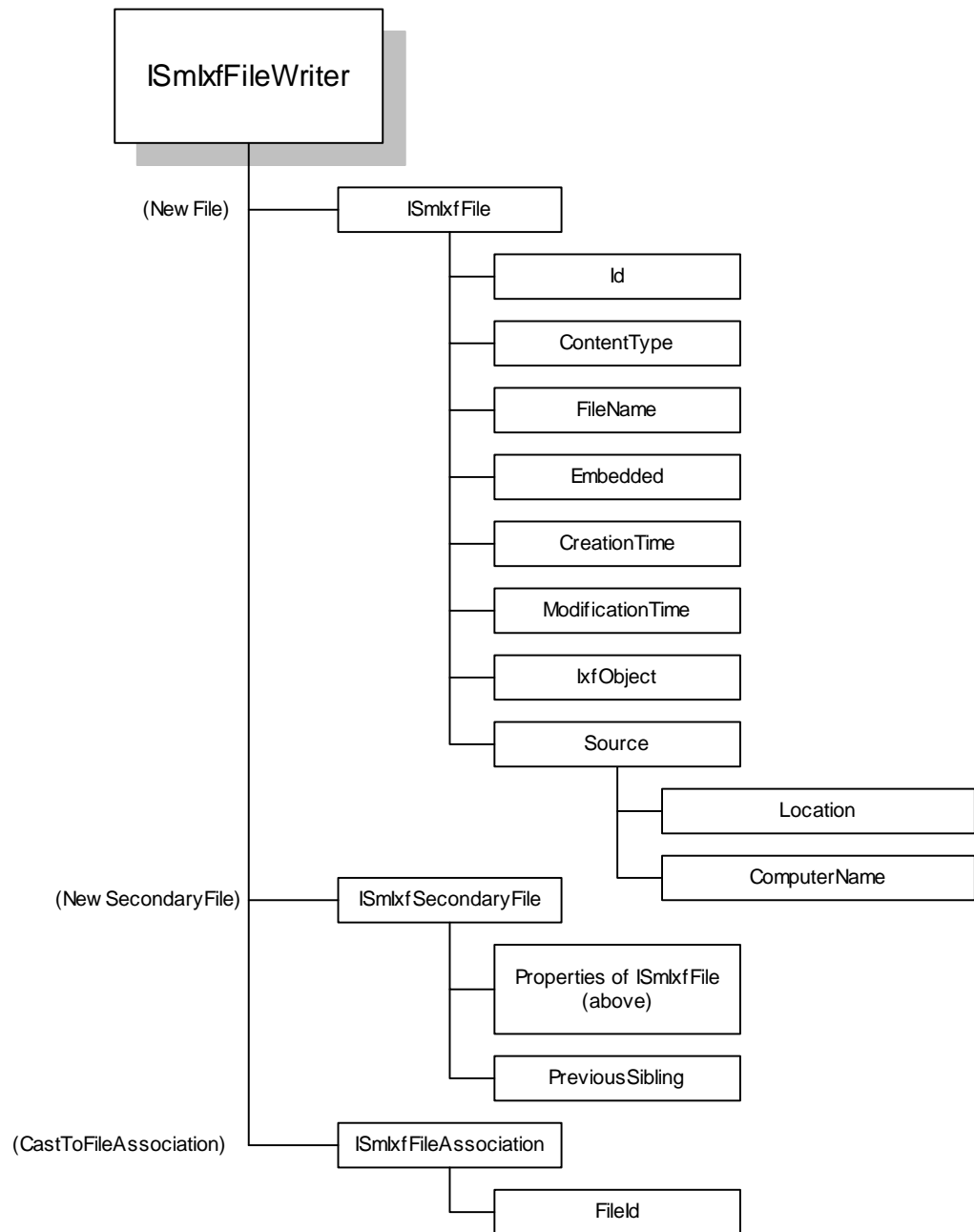


Figure 2-9 ISmxfFileWriter Object Diagram

Obtaining the ISmIxfFileWriter Object

Create a ISmIxfFileWriter object as follows:

```
Dim FileWriter as ISmIxfFileWriter  
Set FileWriter = WriterHelper.FileWriter
```

Methods

The ISmIxfFileWriter object provides methods to write file information to the data file:

Method	Description
NewFile	Creates a new ISmIxfFile object
NewSecondaryFile	Creates a new ISmIxfSecondaryFile object
EmbedFile	Embeds a file in an IXF Archive file.
EmbedSecondaryFile	Embeds an associated file in an IXF Archive file.
CastToFile	Converts a SmIxfObject to a SmIxfFile object. The SmIxfObject to be associated with a file must support the File Association Standard Behavior, otherwise the method returns null.
CastToSecondaryFile	Converts a SmIxfObject to a SmIxfSecondaryFile object. The SmIxfObject to be associated with a file must support the File Association Standard Behavior, otherwise the method returns null.
CastToFileAssociation	Converts an SmIxfObject to a SmIxfFileAssociation object, which is used to associate the object with a file. The SmIxfObject to be associated with a file must support the File Association Standard Behavior, otherwise the method returns null.

Note: In order to use the methods CastTo... you need to have enabled the File Association Standard Behavior support for the class that the IxfObject instantiates, by using the method EnableFileAssociationForClass (see File Association Standard Behavior.)

ISmIxfFile

An ISmIxfFile object represents a primary IXF file and contains its information (see also ISmIxfSecondaryFile).

Obtaining a ISmIxfFile Object

To create a new ISmIxfFile object:

```
Dim File as ISmIxfFile  
Set File = FileWriter.NewFile(Id)
```

Note: ISmIxfFile object can also be obtained through ISmIxfFileReader Object. See FileReader.

Save the file object to the data file after creating it, optionally embedding the physical file into the IXF Archive:

```
FileWriter.EmbedFile(File)  
File.Save
```

where, if used, the EmbedFile method needs to be called prior to the Save method. The File parameter is an existing ISmIxfFile object.

Note: The Save action itself does not embed the physical file to the iXF Archive file.

For an example of how to write and embed a file, see Common Tasks, ISmIxfFileWriter: Writing and embedding a file

Properties

The ISmIxfFile object has the properties:

Property	Description
Id	Input string that uniquely identifies the file object within the IXF Instance Document. Must be a valid NCName.
FileName	Specifies the name of the file
ContentType	The file MIME content type
CreationTime	The file creation time (TDateTime)
ModificationTime	The file last modification time (TDateTime)
Embedded	Indicates whether or not the file is embedded in the iXF archive
IxfObject	Pointer to the ISmIxfObject that is wrapped by the current ISmIxfFile object
Source	The physical location of the file (ISmIxfSource)

ISmIxfSecondaryFile

The ISmIxfSecondaryFile object is provided to handle sequences of files. It represents any member of a sequence of files that is not the first file. The position of a ISmIxfSecondaryFile object in the sequence is determined by its “PreviousSibling” property, which is the Id of the previous file in the sequence.

The ISmIxfSecondaryFile object has the same set of properties as the ISmIxfFile object, shown above, except for the addition of the “PreviousSibling” property.

For example, a sequence of three files might be used to contain the information from one scanned picture.

```
File1 (ISmIxfFile) - PreviousSibling = ""  
File2 (ISmIxfSecondaryFile) - PreviousSibling = "File1"  
File3 (ISmIxfSecondaryFile) - PreviousSibling = "File2"
```

The ISmIxfSecondaryFile object represents the information about a secondary file that is written to the data file.

Creating a New Secondary File:

```
Dim SecondaryFile as ISmIxfSecondaryFile
Set SecondaryFile = FileWriter.NewSecondaryFile(Id, PreviousSibling)
```

Note: ISmIxfSecondaryFile object can also be obtained through ISmIxfFileReader Object. See [FileReader](#).

The file object has to be saved to the data file after finished creating it:

```
FileWriter.EmbedSecondaryFile(SecondaryFile)
SecondaryFile.Save
```

where you need to call the EmbedSecondaryFile method prior to calling the Save method; the file parameter is an existing ISmIxfSecondaryFile object.

Note: The save action does not embed the physical file to the iXF Archive file.

ISmIxfFileAssociation

The ISmIxfFileAssociation object represents an iXF File Association class behavior, which supports associating an iXF Object, such as a Document with a specific file.

The association between file and object is established between a FileAssociation object, which is created from the IxfObject, and the ISmIxfFile object that represents the file.

The object that is to be associated with a file is cast into an object of type ISmIxfFileAssociation using the FileWriter method CastToFileAssociation. The link from object to file is provided through the FileId property of the ISmIxfFileAssociation object, which is set to the Id of the ISmIxfFile object.

Obtaining a FileAssociation Object

```
Dim FileAssociation as ISmIxfFileAssociation
Set FileAssociation = FileWriter.CastToFileAssociation(IxfObject)
```

Properties

The ISmIxfFileAssociation object has one property: the Id of the associated file object, which provides the association between object and file.

For an example of writing a file, see Common Tasks, ISmIxfFileWriter: Writing and embedding a file

ISmIxfVersioningWriter

The ISmIxfVersioningWriter provides the ability to add versioning information to an ISmIxfObject. The ISmIxfVersioningWriter is useful when you need to identify successive revisions of the same entity, for example, successive versions of the same document.

The versioning information itself is represented by an ISmIxfVersioning object, which is obtained from the ISmIxfObject for which versioning is desired by the VersioningWriter method CastToVersioning. The ISmIxfVersioning object includes the version identifiers of the current version and previous version of the entity.

The ISmIxfVersioningWriter has one method: CastToVersioning, which converts an ISmIxfObject to ISmIxfVersioning

Note: In order to use the methods of the ISmIxfVersioningWriter you need to have added the Versioning Standard Behavior support in the schema, using the SchemaHelper method AddDefaultVersioningSupport (see [Versioning Standard Behavior](#))

Obtaining the ISmIxfVersioningWriter Object

```
Dim VersioningWriter as ISmIxfVersioningWriter
Set VersioningWriter = WriterHelper.VersioningWriter
```

ISmIxfVersioning

The ISmIxfVersioning object represents the versioning information for the IxfObject from which it was cast.

Obtaining a Versioning object

```
Dim Versioning as ISmIxfVersioning
Set Versioning = VersioningWriter.CastToVersioning(IxfObject)
```

Note: In order to use the method CastToVersioning on an IxfObject, you need to have enabled the Versioning Standard Behavior support for the class that IxfObject instantiates, using the SchemaHelper method EnableVersioningForClass (see [Versioning Standard Behavior](#))

Properties

The ISmIxfVersioning object has two properties:

Property	Description
PreviousVersion	The previous version identifier of the IxfObject from which this SmIxfVersioning object was cast.
Version	The current version identifier of the IxfObject from which this SmIxfVersioning object was cast

For an example of writing a change, see [Common Tasks](#),
ISmIxfVersioningWriter: Versioning an object

ISmIxfTimeStampWriter

The ISmIxfTimeStampWriter provides the ability to add TimeStamp information to an IxfObject, enabling you to mark the time of object creation and modification.

The TimeStamp information itself is represented by an ISmIxfTimeStamp object, which is obtained from the ISmIxfObject for which time-stamping is desired by the TimeStampWriter method CastToTimeStamp. The ISmIxfTimeStamp object includes the creation time and modification time of the ISmIxfObject.

Note: In order to use the methods of the ISmIxfTimeStampWriter you need to have added the TimeStamp Standard Behavior support in the schema, using the SchemaHelper method AddDefaultTimeStampSupport (see [TimeStamp Standard Behavior](#))

Methods

The ISmIxfTimeStampWriter object has one method: CastToTimeStamp, which converts an ISmIxfObject to ISmIxfTimeStamp

Obtaining the ISmIxfTimeStampWriter Object

```
Dim TimeStampWriter as ISmIxfTimeStampWriter  
Set TimeStampWriter = WriterHelper.TimeStampWriter
```

ISmIxfTimeStamp

The ISmIxfTimeStamp object represents the TimeStamp information for the IxfObject from which it was cast.

Obtaining a ISmIxfTimeStamp Object

```
Dim TimeStamp as ISmIxfTimeStamp  
Set TimeStamp = TimeStampWriter.CastToTimeStamp(IxfObject)
```

Note: In order to use the method CastToTimeStamp you need to have enabled the TimeStamp Standard Behavior support for the class that IxfObject instantiates, using the SchemaHelper method EnableTimeStampForClass (see [TimeStamp Standard Behavior](#))

Properties

The ISmIxfTimeStamp object has two properties:

Property	Description
CreationTime	Time of creation of IxfObject.
ModificationTime	Time of modification of IxfObject.

For an example of writing a change, see Common Tasks, ISmIxfTimeStampWriter: Time-stamping an object

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to the Standard Behavior writers.

ISmIxfChangesWriter: Writing a change

```
Dim StdHelper as ISmIxfStdHelper
Dim WriterHelper as ISmIxfWriterHelper
Dim ChangeWriter as ISmIxfChangeWriter
Dim DocumentObject as Object

'Create stdHelper:
StdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")

'Create Writer Helper:
Set WriterHelper = StdHelper.CreateWriterHelper(IxfWriter)

'Create Change Writer:
Set ChangeWriter = WriterHelper.ChangeWriter

'Create ChangeTransaction:
Set Transaction = ChangeWriter.NewTransaction("1")
Transaction.Save

'Create a new object:
Set IxfObject = IxfWriter.DataWriter.ObjectWriter.NewObject("Document", "3");
...
IxfObject.Save;

'Create a Change for the created object:
Set Change = ChangeWriter.NewChange("2"; ctObjectCreated; "1")
Set Change.ObjectCreated.CreatedObjectRef = "3"
Change.Save
```

ISmIxfFileWriter: Writing and embedding a file

```
Dim StdHelper as ISmIxfStdHelper
Dim WriterHelper as ISmIxfWriterHelper
Dim IxfWriter as ISmIxfWriter
Dim FileWriter as ISmIxfFileWriter
Dim File as ISmIxfFile

'Create stdHelper:
StdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")

'Create a writer:
IxfWriter = CreateObject("SmartIXF1.SmIxfWriter")

'Create Writer Helper:
Set WriterHelper = StdHelper.CreateWriterHelper(IxfWriter)

'Create File Writer:
Set FileWriter = WriterHelper.FileWriter

'Create File:
Set File = FileWriter.NewFile("1")
File.FileName = "design.doc"
File.SetSource = "c:\MyDocuments\design.doc"
FileWriter.EmbedFile(File)
File.Save
```

ISmIxfFileWriter: Associating an object with a file

```
Dim FileAssociation as ISmIxfFileAssociation
Dim DocumentObject as ISmIxfObject

'Create a user-defined object:
Set DocumentObject = IxfWriter.DataWriterObjectWriter.newObject("DocumentMaster",
"2")
.....

'Cast the user-defined object to a file association object:
FileAssociation = FileWriter.CastToFileAssociation(DocumentObject)
FileAssociation.FileId = "1"
```


ISmIxfVersioningWriter: Versioning an Object

```
Dim StdHelper as ISmIxfStdHelper
Dim WriterHelper as ISmIxfWriterHelper
Dim IxfWriter as ISmIxfWriter
Dim VersioningWriter as ISmIxfVersioningWriter
Dim Versioning as ISmIxfVersioning
Dim DocumentObject as ISmIxfObject

'Create stdHelper:
StdHelper = CreateObject("SmartIXFl.SmIxfStdHelper")

'Create a writer:
IxfWriter = CreateObject("SmartIXFl.SmIxfWriter")

'Create Writer Helper:
Set WriterHelper = StdHelper.CreateWriterHelper(IxfWriter)

'Create Versioning Writer:
Set VersioningWriter = WriterHelper.VersioningWriter

'Create a user-defined object:
Set DocumentObject = IxfWriter.DataWriterObjectWriter.NewObject("DocumentMaster",
"4")
.....

'Cast the user-defined object to versioning object:
Versioning = VersioningWriter.CastToVersioning(DocumentObject)
Versioning.Version = "2.0"
Versioning.PreviousVersion = "1.0"
```

ISmIxfTimeStampWriter: Time-stamping an object

```
Dim StdHelper as ISmIxfStdHelper
Dim WriterHelper as ISmIxfWriterHelper
Dim TimeStampWriter as ISmIxfTimeStampWriter

'Create stdHelper:
StdHelper = CreateObject("SmartIXFl.SmIxfStdHelper")

'Create Writer Helper:
Set WriterHelper = StdHelper.CreateWriterHelper(IxfWriter)
Set TimeStampWriter = WriterHelper.TimeStampWriter

'Create TimeStamp:
TimeStamp = TimeStampWriter.CastToTimeStamp(DocumentObject)
```

```
TimeStamp.CreationTime = now
```

ISmIxfReaderHelper

The ISmIxfReaderHelper object supports reading Standard Behaviors information from a data file.

Identifying and Restoring Read-In Objects

The Standard Behaviors object data is written by the DataWriter to the IXF Instance data file as the original object types such as ISmIxfChange, ISmIxfFile, ISmIxfVersioning, as described in the section on the ISmIxfWriterHelper.

However, the same object data is read by the DataReader from the IXF Instance data file as generic ISmIxfObject objects rather than as the object types that were originally written to the data file. To identify and restore the original object types, the ReaderHelper provides methods for casting the ISmIxfObject objects read from the data file back to the same type of objects that were written.

For each object read in, you need to run all the cast methods on it successively. When a specific cast method returns a non-null result, you have identified and restored the object.

Object Diagram

The object diagram of ISmIxfReaderHelper is shown below:

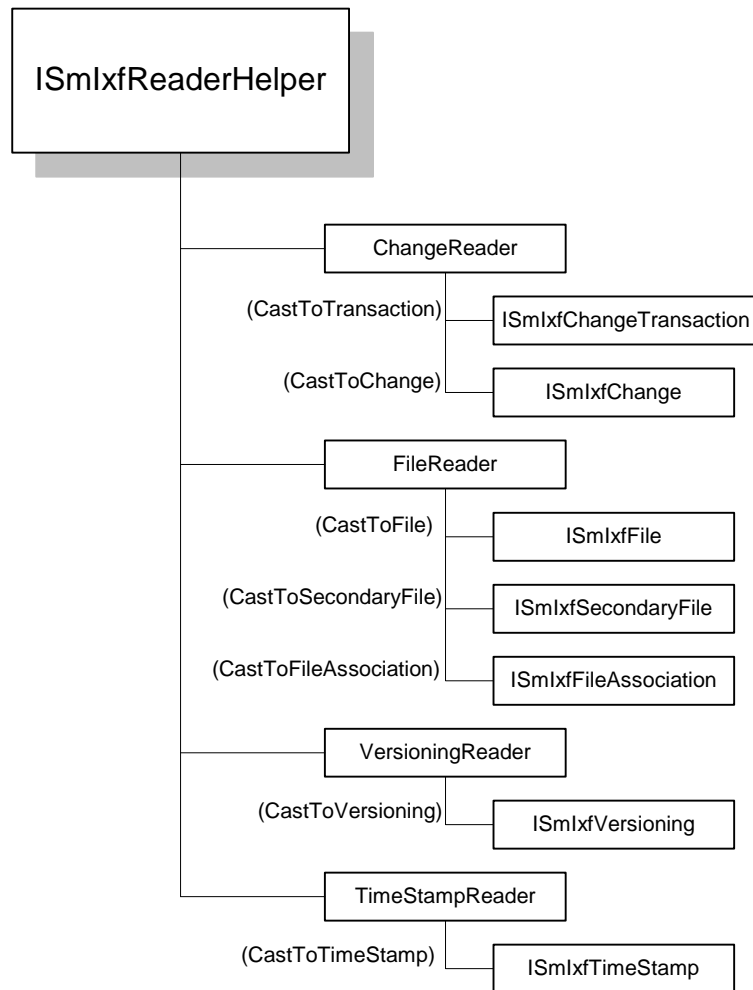


Figure 2-10 *ISmIxfReaderHelper* Object Diagram

Properties

Four ReaderHelper properties are provided corresponding to the supported Standard Behaviors:

Property	Description
ChangeReader	Provides methods for reading Change-Tracking objects
FileReader	Provides methods for reading File Association objects
VersioningReader	Provides methods for reading Versioning objects
TimeStampReader	Provides methods for reading TimeStamp objects

Obtaining the ISmIxfReaderHelper Object

To obtain an ISmIxfReaderHelper Object:

```
Dim ReaderHelper as ISmIxfReaderHelper
Set ReaderHelper = StdHelper.CreateReaderHelper(IxfReader)
```

ChangeReader

The ChangeReader helps to read the change-tracking objects ISmIxfChange and ISmIxfChangeTransaction from the Ixf Archive data file. The ChangeReader identifies and restores the original object types, by casting the ISmIxfObject objects read from the Ixf Archive data file back to the same type of objects that were written.

Obtaining the ISmIxfChangeReader Object

```
Dim ChangeReader as ISmIxfChangeReader
Set ChangeReader = ReaderHelper.ChangeReader
```

Methods

The ISmIxfChangeReader object provides the following methods to read Change-Tracking information from the data file.

Method	Description
CastToChange	Converts an ISmIxfObject to ISmIxfChange
CastToTransaction	Converts an ISmIxfObject to ISmIxfChangeTransaction

Note: In order to use these methods, you need to have added the Change-Tracking Standard Behavior support in the schema, using the SchemaHelper method AddDefaultChangesSupport (see [Change-Tracking Standard Behavior](#))

ISmIxfChangeTransaction

To cast an object to a ChangeTransaction object:

```
Dim Transaction as ISmIxfChangeTransaction
Set Transaction = ChangeReader.CastToTransaction(IxfObject)
```

See ISmIxfChangeTransaction section under [ISmIxfChangeWriter](#) for details about the ISmIxfChangeTransaction properties.

ISmIxfChange

To cast an object to a Change object:

```
Dim Transaction as ISmIxfChangeTransaction
Set Transaction = ChangeReader.CastToTransaction(IxfObject)
```

See ISmIxfChange section under [ISmIxfChangeWriter](#) section for details about the ISmIxfChange properties.

For an example of how to use ISmIxfReaderHelper to read Change objects, see [Common Tasks](#), Reading and Casting objects to File and to Change objects

FileReader

The FileReader helps to read the File Association objects ISmIxfFile, ISmIxfSecondaryFile and ISmIxfFileAssociation from the Ixf Archive data file. The FileReader identifies and restores the original object types, by casting the ISmIxfObject objects read from the Ixf Archive data file back to the same type of objects that were written.

Obtaining the ISmIxfFileReader Object

To obtain the ISmIxfFileReader object:

```
Dim FileReader as ISmIxfFileReader  
Set FileReader = ReaderHelper.FileReader
```

Methods

The ISmIxfFileReader object provides the following methods for reading file information from the data file:

Method	Description
CastToFile	Converts an ISmIxfObject to ISmIxfFile
CastToSecondaryFile	Converts an ISmIxfObject to ISmIxfSecondaryFile
CastToFileAssociation	Converts an ISmIxfObject to an ISmIxfFileAssociation object

Note: In order to use these methods, you need to have added the File Association Standard Behavior support in the schema by including the AddDefaultFilesSupport method in the schema (see [File Association Standard Behavior](#).) In addition, you need to have enabled the File Association Standard Behavior support for the class that IxfObject instantiates, by including the method EnableFileAssociationForClass (see [File Association Standard Behavior](#).)

ISmIxfFile

The ISmIxfFile object represents a primary IXF file and contains the file information.

To cast an object to File object:

```
Dim File as ISmIxfFile
Set File = FileReader.CastToFile(IxfObject)
```

See ISmIxfFile section for details about the ISmIxfFile properties.

In order to extract an embedded file from the iXF Archive file, you can use one the following methods of ISmIxfFile object:

```
File.Extract(RootFolder)
File.ExtractToFile(NewFileName)
```

See the reference guide for more details about those functions.

For an example of how to use ISmIxfReaderHelper to read File objects, see [Common Tasks](#), Reading and Casting objects to File and to Change objects

ISmIxfSecondaryFile

To cast an object to a SecondaryFile object:

```
Dim SecondaryFile as ISmIxfSecondaryFile
Set SecondaryFile = FileReader.CastToSecondaryFile(Id, PreviousSibling)
```

See ISmIxfFile section for details about the ISmIxfSecondaryFile properties.

ISmIxfFileAssociation

To cast an object to a FileAssociation object:

```
Dim FileAssociation as ISmIxfFileAssociation
Set FileAssociation = FileWriter.CastToFileAssociation(IxfObject)
```

See ISmIxfFileAssociation for information about the ISmIxfFileAssociation properties.

VersioningReader

The VersioningReader helps to read ISmIxfVersioning objects from the Ixf Archive data file. The VersioningReader identifies and restores the original object types, by casting the ISmIxfObject objects read from the Ixf Archive data file back to the same type of objects that were written.

Obtaining the ISmIxfVersioningReader Object

To obtain a ISmIxfVersioningReader Object:

```
Dim VersioningReader as ISmIxfVersioningReader  
Set VersioningReader = ReaderHelper.VersioningReader
```

Methods

The ISmIxfVersioningReader has one method: CastToVersioning, which converts an ISmIxfObject to a ISmIxfVersioning object.

Note: In order to use this method, you need to have added the Versioning Standard Behavior support in the schema, using the SchemaHelper method AddDefaultVersioningSupport (see Versioning Standard Behavior). In addition, to use CastToVersioning on an IxfObject, you need to have enabled the Versioning Standard Behavior support for the class that IxfObject instantiates, by including the method EnableVersioningForClass (see File Association Standard Behavior.)

ISmIxfVersioning

See the ISmIxfVersioning section under ISmIxfVersioningWriter for information about this object.

TimeStampReader

The TimeStampReader helps to read ISmIxfTimeStamp objects from the Ixf Archive data file. The TimeStampReader identifies and restores the original object types, by casting the ISmIxfObject objects read from the Ixf Archive data file back to the same type of objects that were written.

Obtaining the ISmIxfTimeStampReader Object

```
Dim TimeStampReader as ISmIxfTimeStampReader  
Set TimeStampReader = WriterHelper.TimeStampReader
```

Methods

The ISmIxfTimeStampReader object has one method: CastToTimeStamp, which converts an ISmIxfObject to a ISmIxfTimeStamp object.

Note: In order to use this method, you need to have added the TimeStamp Standard Behavior support in the schema, using the SchemaHelper method AddDefaultTimeStampSupport (see [TimeStamp Standard Behavior](#)). In addition, to use CastToTimeStamp on an IxfObject, you need to have enabled the TimeStamp Standard Behavior support for the class that IxfObject instantiates, by including the method EnableTimeStampForClass (see [TimeStamp Standard Behavior](#))

ISmIxfTimeStamp

See ISmIxfTimeStamp section under the ISmIxfTimeStampWriter object for information about this object.

Common Tasks

The following sections describe methods and properties that are used to perform common tasks related to an ISmIxfReaderHelper.

ISmIxfReaderHelper:

Reading and Casting objects to File and to Change objects

```
Dim StdHelper As ISmIxfStdHelper
Dim ReaderHelper As ISmIxfReaderHelper
Dim ObjectIterator As ISmIxfObjectIterator
Dim IxfObject As ISmIxfObject
Dim Change As ISmIxfChange
Dim ChangeId, CreatedObjectId As Sting
Dim File As ISmIxfFile

'Create stdHelper:
StdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")

'Create Reader Helper:
Set ReaderHelper = StdHelper.CreateReaderHelper(IxfReader)

'Create ObjectIterator:
Set ObjectIterator = IxfReader.DataReader.ObjectReader.GetObjectIterator

While ObjectIterator.AtEnd = False
    Set IxfObject = ObjectIterator.GetObject
```

```

    'Read Change
    Change = ReaderHelper.ChangeReader.CastToChange(IxfObject)
    If Not (Change Is Nothing) Then
        ChangeId = Change.Id
        If Change.ChangeType = ctObjectCreated Then
            CreatedObjectId = Change.ObjectCreated.CreatedObjectRef
        End If
        ...
    End If

    'Read File
    File = ReaderHelper.FileReader.CastToFile(IxfObject)
    If Not (File Is Nothing) Then
        FileName = File.FileName
        File.Extract("Ixf Sample")
        ...
    End If
    ObjectIterator.Next
Wend

```

3 Sample IXF Application

This chapter presents a sample iXF application, which demonstrates many of the objects, properties and methods described above.

The sample application includes generating and processing an iXF package for a messaging application, using the basic SmartIxf1.0 API functionality. This example is included in the SDK under Samples/SmartIxf/Vb/Sample1.

A simple messaging format is defined, which includes the basic messaging entities: message, attachment and folder.

Messaging Format

Entity	Attributes
Message	<ul style="list-style-type: none">- From- To- Subject- Body- Importance- Time of sending
Folder	<ul style="list-style-type: none">- Name- Creation time
Attachment	<ul style="list-style-type: none">- File reference
FolderLink	<ul style="list-style-type: none">- Parent folder- Child folder
FolderMessageLink	<ul style="list-style-type: none">- Parent folder- Child message
MessageAttachmentLink	<ul style="list-style-type: none">- Parent message- Child attachment

Class Behaviors

The following table lists the ClassBehaviors.

ClassBehavior	URI	Attributes
Message	<code><exemplens-c></code> <code>/messaging#message</code>	From: String, required To: String, required Subject: String, not required Body: String, not required Importance: Integer, not required, default value = 0
Folder	<code><exemplens-c></code> <code>/messaging#folder</code>	Name: String, required
Link	<code><ixfstdns-c></code> <code>/links/1.0#link</code>	Object1, Object2
Directed Link	<code><ixfstdns-c></code> <code>/links/1.0#directedLink</code>	Object1, Object2 Directed from Object1 to Object2
Tree Link	<code><ixfstdns-c></code> <code>/links/1.0#treeLink</code>	Object1, Object2 Object1 is the only parent of Object2
TimeStamp	<code><ixfstdns-c></code> <code>/timeStamp/1.0#timeStamp</code>	creationTime modificationTime
FileAssociation	<code><ixfstdns-c></code> <code>/files/1.0#fileAssociation</code>	file

The following abbreviations are used in the table:

<code><exemplens-c></code>	http://example.com/classBehaviors
<code><ixfstdns-c></code>	http://www.ixfstd.org/std/ns/core/classBehaviors

Domain Behaviors

This section describes the Domain Behavior defined for the messaging application. The URI for the Domain Behavior is:

<http://example.com/domainBehaviors/messaging>.

Domain Behavior Definition

The following table defines the Roles and, for each Role, the Class Behaviors that must be implemented by the class, which is mapped to the role. The timeStamp Standard Behavior is only included in the Message and Folder Roles.

Role	Required Class Behaviors
Message	<exemplens-c>/messaging#message <ixfstdns-c>/timeStamp/1.0#timeStamp
Folder	<exemplens-c>/messaging#folder <ixfstdns-c>/timeStamp/1.0#timeStamp
FolderLink	<ixfstdns-c>/links/1.0#link <ixfstdns-c>/links/1.0#directedLink <ixfstdns-c>/links/1.0#treeLink Informal restriction: must point to folder-behavior objects
Attachment	<ixfstdns-c>/files/1.0#fileAssociation
MessageAttachmentLink	<ixfstdns-c>/links/1.0#link <ixfstdns-c>/links/1.0#directedLink informal restriction: parent = message, child = attachment
FolderMessageLink	<ixfstdns-c>/links/1.0#link <ixfstdns-c>/links/1.0#directedLink informal restriction: parent = folder, child = message

Role-to-Class Mapping

The Role-to-Class mapping for the Domain Behavior is:

Role	Class
Message	Message
Folder	Folder
FolderLink	FolderLink
Attachment	Attachment
AttachmentLink	AttachmentLink
FolderMessageLink	FolderMessageLink

Connectivity of Objects

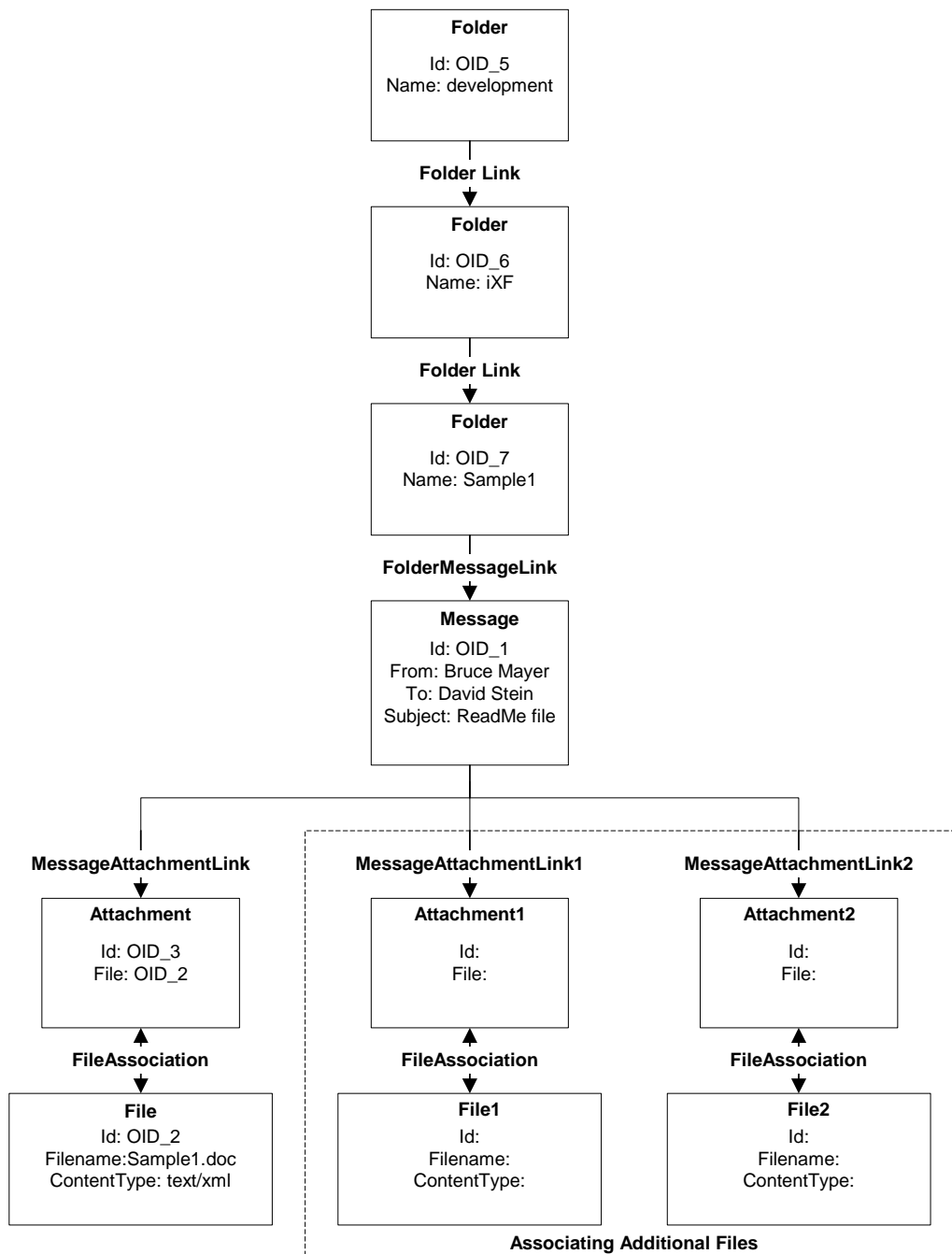
The following diagram shows the connectivity of the basic object and the link objects in the example.

Associating Files with Messages

Note that although the file is associated with the message, the File object is not associated directly to the Message object, but rather through an Attachment object. The File is associated with the Attachment object through the FileAssociation Standard Behavior and the attachment object is linked to the message object with the MessageAttachmentLink.

The reason it is done this way is that the FileAssociation Standard Behavior allows you to associate at most one file with an object enabled for FileAssociation. Thus, to allow for the possibility of associating more than one file to a message, the messaging application has been designed with the intermediate Attachment object and the MessageAttachmentLink object. For each file you want to associate with a message, you create a separate Attachment object and a corresponding MessageAttachmentLink object and follow the procedure of the example.

The figure below shows how you would associate more than one file to the message.



Implementing the Application

This section shows code examples of how the messaging application is implemented. This section does not include all the code in the example, but rather the code needed to illustrate and explain the implementation. For the full code, see the example included in:
SDK/Samples/SmartIxf/Vb/Sample1/Sample1.vbp

Creating the Schema

This section shows how to create the schema for the application and includes the topics:

- Adding Class Behaviors
- Adding Classes
- Adding Domain Behaviors

Adding Class Behaviors

The following functions add the required Class Behaviors to the schema:

Add API-Supported ClassBehaviors

```
Private Sub (Schema As ISmIxfSchema, SchemaHelper As ISmIxfSchemaHelper)
    'Add iXF TimeStamp Standard Class Behavior
    SchemaHelper.AddDefaultTimeStampSupport

    'Add iXF FileAssociation Standard Class Behavior
    SchemaHelper.AddDefaultFilesSupport
End Sub
```

Add Message ClassBehavior

```
Private Sub AddMessageClassBehavior(Schema As ISmIxfSchema)
    Dim IxfClassBehavior As ISmIxfClassBehavior
    Dim IxfAttribute As ISmIxfAttribute

    'Add Message ClassBehavior to Schema ClassesBehaviors
    Set IxfClassBehavior = Schema.ClassesBehaviors.Add(mtEmbedded,
    CB_MESSAGE_URI)

    'Add "from" attribute to Message ClassBehavior
```

```

Set IxfAttribute = IxfClassBehavior.Attributes.Add("from")
IxfAttribute.TypeDefinition.ValueType = dtString
IxfAttribute.Required = True

'Add "to" attribute to Message ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("to")
IxfAttribute.TypeDefinition.ValueType = dtString
IxfAttribute.Required = True

'Add "subject" attribute to Message ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("subject")
IxfAttribute.TypeDefinition.ValueType = dtString
IxfAttribute.Required = False

'Add "body" attribute to Message ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("body")
IxfAttribute.TypeDefinition.ValueType = dtString
IxfAttribute.Required = False

'Add "importance" attribute to Message ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("importance")
IxfAttribute.TypeDefinition.ValueType = dtInt
IxfAttribute.Required = False
IxfAttribute.DefaultValue = 0
End Sub

```

Add Folder ClassBehavior

```

Private Sub AddFolderClassBehavior(Schema As ISmIxfSchema)
    Dim IxfClassBehavior As ISmIxfClassBehavior
    Dim IxfAttribute As ISmIxfAttribute

    'Add Folder ClassBehavior to Schema ClassesBehaviors
    Set IxfClassBehavior = Schema.ClassesBehaviors.Add(mtEmbedded,
CB_FOLDER_URI)

    'Add "name" attribute to Folder ClassBehavior
    Set IxfAttribute = IxfClassBehavior.Attributes.Add("name")
    IxfAttribute.TypeDefinition.ValueType = dtString
    IxfAttribute.Required = True
End Sub

```

Add Link ClassBehavior

```

Private Sub AddLinkClassBehavior(Schema As ISmIxfSchema)

```

```
Dim IxfClassBehavior As ISmIxfClassBehavior
Dim IxfAttribute As ISmIxfAttribute

'Add IXF Standard ClassBehavior "Link" to Schema ClassesBehaviors
Set IxfClassBehavior = Schema.ClassesBehaviors.Add(mtEmbedded,
CB_LINK_URI)

'Add "object1" attribute to Link ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("object1")
IxfAttribute.TypeDefinition.ValueType = dtObjectReference
IxfAttribute.Required = True

'Add "object2" attribute to Link ClassBehavior
Set IxfAttribute = IxfClassBehavior.Attributes.Add("object2")
IxfAttribute.TypeDefinition.ValueType = dtObjectReference
IxfAttribute.Required = True
End Sub
```

To add Directed Link, Tree Link, and TimeStamp Class Behaviors, see source files.

Execute Functions

```
Public Sub AddClassBehaviorsDefinitionsToSchema(Schema As ISmIxfSchema,
SchemaHelper As ISmIxfSchemaHelper)
    AddAPISupportedClassBehaviors Schema, SchemaHelper
    AddMessageClassBehavior Schema
    AddFolderClassBehavior Schema
    AddLinkClassBehavior Schema
    AddDirectedLinkClassBehavior Schema
    AddTreeLinkClassBehavior Schema
End Sub
```

Adding Classes

The following functions add the required classes to the schema:

Add Message Class

```
Private Sub AddMessageClass(Schema As ISmIxfSchema, SchemaHelper As
ISmIxfSchemaHelper)
    Dim IxfClass As ISmIxfClass
    Dim IxfClassBehavior As ISmIxfClassBehavior

    'Add class "message" to Schema Classes
    Set IxfClass = Schema.Classes.Add("message")
```

```

        'Enable IXF TimeStamp Standard Behavior for message class as
        required
        'by DB_MESSAGING_URI Domain Behavior
        SchemaHelper.EnableTimeStampForClass IxfClass

        'Declare CB_MESSAGE_URI ClassBehavior in message class as required
        'by DB_MESSAGING_URI Domain Behavior
        Set IxfClassBehavior =
        Schema.ClassesBehaviors.ItemByURI(CB_MESSAGE_URI)
        IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
    End Sub

```

Add Folder Class

```

Private Sub AddFolderClass(Schema As ISmIxfSchema, SchemaHelper As
ISmIxfSchemaHelper)
    Dim IxfClass As ISmIxfClass
    Dim IxfClassBehavior As ISmIxfClassBehavior

    'Add class "folder" to Schema Classes
    Set IxfClass = Schema.Classes.Add("folder")

    'Enable IXF TimeStamp Standard Behavior for folder class as required
    'by DB_MESSAGING_URI Domain Behavior
    SchemaHelper.EnableTimeStampForClass IxfClass

    'Declare CB_FOLDER_URI ClassBehavior in folder class as required
    'by DB_MESSAGING_URI Domain Behavior
    Set IxfClassBehavior =
    Schema.ClassesBehaviors.ItemByURI(CB_FOLDER_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
End Sub

```

Add FolderLink Class

```

Private Sub AddFolderLinkClass(Schema As ISmIxfSchema)
    Dim IxfClass As ISmIxfClass
    Dim IxfClassBehavior As ISmIxfClassBehavior

    'Add class "folderLink" to Schema Classes
    Set IxfClass = Schema.Classes.Add("folderLink")

    'Declare CB_LINK_URI, CB_DIRECTEDLINK_URI, and CB_TREELINK_URI
    'ClassBehaviors in folderLink class as required
    'by DB_MESSAGING_URI Domain Behavior. When writing the object, only

```

```

        'the CB_LINK_URI is used. (The presence of the CB_DIRECTEDLINK_URI,
        'and CB_TREELINK_URI ClassBehaviors cause the CB_LINK_URI to be
        'interpreted as a directed parent-son tree link.)
    Set IxfClassBehavior = Schema.ClassesBehaviors.ItemByURI(CB_LINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
    Set IxfClassBehavior =
    Schema.ClassesBehaviors.ItemByURI(CB_DIRECTEDLINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
    Set IxfClassBehavior =
    Schema.ClassesBehaviors.ItemByURI(CB_TREELINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
End Sub

```

Add Attachment Class

```

Private Sub AddAttachmentClass(Schema As ISmIxfSchema, SchemaHelper As
ISmIxfSchemaHelper)
    Dim IxfClass As ISmIxfClass

    'Add class "attachment" to Schema Classes
    Set IxfClass = Schema.Classes.Add("attachment")
    'Enable IXF FileAssociation Standard Behavior for folder class as
    'required by DB_MESSAGING_URI Domain Behavior
    SchemaHelper.EnableFileAssociationForClass IxfClass
End Sub

```

Add MessageAttachmentLink Class

```

Private Sub AddMessageAttachmentLinkClass(Schema As ISmIxfSchema)
    Dim IxfClass As ISmIxfClass
    Dim IxfClassBehavior As ISmIxfClassBehavior

    'Add class "messageAttachmentLink" to Schema Classes
    Set IxfClass = Schema.Classes.Add("messageAttachmentLink")

    'Declare CB_LINK_URI, and CB_DIRECTEDLINK_URI ClassBehaviors in
    'messageAttachmentLink class as required by DB_MESSAGING_URI Domain
    'Behavior
    Set IxfClassBehavior = Schema.ClassesBehaviors.ItemByURI(CB_LINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
    Set IxfClassBehavior =
    Schema.ClassesBehaviors.ItemByURI(CB_DIRECTEDLINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
End Sub

```

Add FolderMessageLink Class

```
Private Sub AddFolderMessageLinkClass(Schema As ISmIxfSchema)
    Dim IxfClass As ISmIxfClass
    Dim IxfClassBehavior As ISmIxfClassBehavior

    'Add class "folderMessageLink" to Schema Classes
    `Set IxfClass = Schema.Classes.Add("folderMessageLink")

    'Declare CB_LINK_URI, and CB_DIRECTEDLINK_URI ClassBehaviors in
    'folderMessageLink class as required by DB_MESSAGING_URI Domain
    'Behavior
    Set IxfClassBehavior = Schema.ClassesBehaviors.ItemByURI(CB_LINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
    Set IxfClassBehavior =
    Schema.ClassesBehaviors.ItemByURI(CB_DIRECTEDLINK_URI)
    IxfClass.CurrentClassBehaviors.Add IxfClassBehavior
End Sub
```

Execute Functions

```
Public Sub AddClassesDefininitionsToSchema(Schema As ISmIxfSchema,
SchemaHelper As ISmIxfSchemaHelper)
    AddMessageClass Schema, SchemaHelper
    AddFolderClass Schema, SchemaHelper
    AddFolderLinkClass Schema
    AddAttachmentClass Schema, SchemaHelper
    AddMessageAttachmentLinkClass Schema
    AddFolderMessageLinkClass Schema
End Sub
```

Adding Domain Behaviors

The following functions add the required Domain Behavior to the schema:

Add Messaging Domain Behavior

```
Private Sub AddMessagingDomainBehavior(Schema As ISmIxfSchema)
    Dim IxfDomainBehavior As ISmIxfDomainBehavior
    Dim IxfClass As ISmIxfClass

    'Add Domain Behavior DB_MESSAGING_URI" to Schema DomainBehaviors
    Set IxfDomainBehavior = Schema.DomainBehaviors.Add(DB_MESSAGING_URI)

    'Assign the Domain Behavior Roles to their corresponding classes
    Set IxfClass = Schema.Classes.ItemByName("message")
```

```
IxfDomainBehavior.RoleClassMapping("message") = IxfClass
Set IxfClass = Schema.Classes.ItemByName("folder")
IxfDomainBehavior.RoleClassMapping("folder") = IxfClass
Set IxfClass = Schema.Classes.ItemByName("folderLink")
IxfDomainBehavior.RoleClassMapping("folderLink") = IxfClass
Set IxfClass = Schema.Classes.ItemByName("attachment")
IxfDomainBehavior.RoleClassMapping("attachment") = IxfClass
Set IxfClass = Schema.Classes.ItemByName("messageAttachmentLink")
IxfDomainBehavior.RoleClassMapping("messageAttachmentLink") = IxfClass
End Sub
```

Execute Functions

```
Public Sub AddDomainBehaviorsToTheSchema(Schema As ISmIxfSchema)
    AddMessagingDomainBehavior Schema
End Sub
```

Writing the Data

This section shows how to write the data to a data file. Two types of objects are written:

- Basic Objects
- Link Objects

Basic Objects

Write Message Object

```
Private Sub CreateMessageObject(DataWriter As ISmIxfDataWriter,
WriterHelper As ISmIxfWriterHelper)
    Dim IxfObject As ISmIxfObject
    Dim BehaviorValues As ISmIxfAttributesValues
    Dim TimeStamp As ISmIxfTimeStamp

    'Instantiate an object from the message class; give it an Id
    Set IxfObject = DataWriter.ObjectWriter.NewObject("message", "OID_1")

    'Assign values to CB_MESSAGE_URI BehaviorValues
    Set BehaviorValues = IxfObject.GetBehaviorValues(CB_MESSAGE_URI)
    BehaviorValues.Item("from") = "Bruce Mayer"
    BehaviorValues.Item("to") = "David Stein"
    BehaviorValues.Item("subject") = " The Sample1.doc file "
    BehaviorValues.Item("body") = " Attached is the Sample1.doc file"
```



```

        'TimeStamp the message object
        Set TimeStamp =
WriterHelper.TimeStampWriter.CastToTimeStamp(IxfObject)
        TimeStamp.CreationTime = Now

        'Save the message object to the data file
        IxfObject.Save
End Sub

```

Create and Embed the File Object

```

Private Sub CreateFileObjectAndEmbedFile(WriterHelper As
ISmIxfWriterHelper)
    Dim IxfFile As ISmIxfFile

    'Instantiate a File object and give it values
    Set IxfFile = WriterHelper.FileWriter.NewFile("OID_2")
    IxfFile.FileName = " Sample1.doc"
    IxfFile.ContentType = "text/xml"
    IxfFile.SetSource ("Sample1.doc")

    'Embed the File object and save it to the data file
    WriterHelper.FileWriter.EmbedFile IxfFile
    IxfFile.Save
End Sub

```

Write Attachment Object

```

Private Sub CreateAttachmentObject(DataWriter As ISmIxfDataWriter,
WriterHelper As ISmIxfWriterHelper)
    Dim IxfObject As ISmIxfObject
    Dim FileAssociation As ISmIxfFileAssociation

    'Instantiate an object from the attachment class; give it an Id
    Set IxfObject = DataWriter.ObjectWriter.NewObject("attachment",
"OID_3")
    'Associate the File object OID_2 with the attachment object
    Set FileAssociation =
WriterHelper.FileWriter.CastToFileAssociation(IxfObject)
    FileAssociation.FileId = "OID_2"

    'Save the attachment object
    IxfObject.Save
End Sub

```

Write Folder Objects

```
Private Sub CreateFolderObjects(DataWriter As ISmIxfDataWriter,
WriterHelper As ISmIxfWriterHelper)

    Dim IxfObject As ISmIxfObject
    Dim BehaviorValues As ISmIxfAttributesValues
    Dim TimeStamp As ISmIxfTimeStamp

    'Development folder
    'Instantiate an object from the folder class; give it an Id
    Set IxfObject = DataWriter.ObjectWriter.NewObject("folder", "OID_5")

    'Get CB_FOLDER_URI ClassBehavior BehaviorValues for this folder
    'object
    Set BehaviorValues = IxfObject.GetBehaviorValues(CB_FOLDER_URI)
    'Name the folder "Development"
    BehaviorValues.Item("name") = "Development"

    'Time-stamp the "Development" folder
    Set TimeStamp =
WriterHelper.TimeStampWriter.CastToTimeStamp(IxfObject)
    TimeStamp.CreationTime = Now

    'Save the "Development" folder
    IxfObject.Save

    'iXF folder
    'Instantiate another object from the folder class; give it an Id
    Set IxfObject = DataWriter.ObjectWriter.NewObject("folder", "OID_6")

    'Get CB_FOLDER_URI ClassBehavior BehaviorValues for this folder
    'object and name it "iXF"
    Set BehaviorValues = IxfObject.GetBehaviorValues(CB_FOLDER_URI)
    BehaviorValues.Item("name") = "iXF"

    'Time-stamp the "iXF" folder
    Set TimeStamp =
WriterHelper.TimeStampWriter.CastToTimeStamp(IxfObject)
    TimeStamp.CreationTime = Now

    'Save the "iXF" folder
    IxfObject.Save

    'Sample1 folder
    'Instantiate another object from the folder class; give it an Id
```

```

Set IxfObject = DataWriter.ObjectWriter.NewObject("folder", "OID_7")

'Get CB_FOLDER_URI ClassBehavior BehaviorValues for this folder
'object and name it "Sample1"
Set BehaviorValues = IxfObject.GetBehaviorValues(CB_FOLDER_URI)
BehaviorValues.Item("name") = "Sample1"

'Time-stamp the "Sample1" folder
Set TimeStamp =
WriterHelper.TimeStampWriter.CastToTimeStamp(IxfObject)
TimeStamp.CreationTime = Now

'Save the "Sample1" folder
IxfObject.Save
End Sub

```

Link Objects

Write MessageAttachmentLink Object

```

Private Sub CreateMessageAttachmentLinkObject(DataWriter As
ISmIxfDataWriter)
    Dim IxfObject As ISmIxfObject
    Dim BehaviorValues As ISmIxfAttributesValues

    'Instantiate an object from the messageAttachmentLink class;
    'give it an Id
    Set IxfObject =
    DataWriter.ObjectWriter.NewObject("messageAttachmentLink", "OID_4")
    'Get CB_LINK_URI ClassBehavior BehaviorValues for this object
    Set BehaviorValues = IxfObject.GetBehaviorValues(CB_LINK_URI)
    'Link between the message object and the attachment object
    BehaviorValues.Item("object1") = "OID_1"
    BehaviorValues.Item("object2") = "OID_3"

    'Save the messageAttachmentLink object
    IxfObject.Save
End Sub

```

Write FolderLink Objects

```

Private Sub CreateFolderLinkObjects(DataWriter As ISmIxfDataWriter)
    Dim IxfObject As ISmIxfObject
    Dim BehaviorValues As ISmIxfAttributesValues

    'Link "Development" folder as parent of "iXF" folder

```

```

        Set IxfObject = DataWriter.ObjectWriter.NewObject("folderLink",
"OID_8")
        Set BehaviorValues = IxfObject.GetBehaviorValues(CB_LINK_URI)
        BehaviorValues.Item("object1") = "OID_5"
        BehaviorValues.Item("object2") = "OID_6"

        'Save the folderLink
        IxfObject.Save

        'Link "iXF" folder as parent of "Sample1" folder
        Set IxfObject = DataWriter.ObjectWriter.NewObject("folderLink",
"OID_9")
        Set BehaviorValues = IxfObject.GetBehaviorValues(CB_LINK_URI)
        BehaviorValues.Item("object1") = "OID_6"
        BehaviorValues.Item("object2") = "OID_7"

        'Save the folderLink
        IxfObject.Save
End Sub

```

Write FolderMessageAttachment Objects

```

Private Sub CreateFolderMessageAttachmentObjects(DataWriter As
ISmIxfDataWriter)
    Dim IxfObject As ISmIxfObject
    Dim BehaviorValues As ISmIxfAttributesValues

    'Link where "Sample1" folder as parent of the message with the
    'subject "The Sample1.doc file"
    Set IxfObject = DataWriter.ObjectWriter.NewObject("folderMessageLink",
"OID_8")
    Set BehaviorValues = IxfObject.GetBehaviorValues(CB_LINK_URI)
    BehaviorValues.Item("object1") = "OID_7"
    BehaviorValues.Item("object2") = "OID_1"

    'Save the link
    IxfObject.Save
End Sub

```

Execute Functions

```

Public Sub CreateData(Writer As ISmIxfWriter, StdHelper As SmIxfStdHelper)
    Dim WriterHelper As ISmIxfWriterHelper
    Dim DataWriter As ISmIxfDataWriter

    Set WriterHelper = StdHelper.CreateWriterHelper(Writer)

```

```

CreateMessageObject Writer.DataWriter, WriterHelper
CreateFileObjectAndEmbedFile WriterHelper
CreateAttachmentObject Writer.DataWriter, WriterHelper
CreateMessageAttachmentLinkObject Writer.DataWriter
CreateFolderObjects Writer.DataWriter, WriterHelper
CreateFolderLinkObjects Writer.DataWriter
CreateFolderMessageAttachmentObjects Writer.DataWriter
End Sub

```

Reading the Data

This section shows how to use the Reader and the ReaderHelper to read the data file.

Read Data Objects

```

Public Sub ReadData(Reader As ISmIxfReader, StdHelper As SmIxfStdHelper)
    Dim ReaderHelper As ISmIxfReaderHelper
    Dim ObjectIterator As ISmIxfObjectIterator
    Dim IxfObject As ISmIxfObject
    Dim IxfFile As ISmIxfFile

    Set ReaderHelper = StdHelper.CreateReaderHelper(Reader)
    Set ObjectIterator = Reader.DataReader.ObjectReader.GetObjectIterator

    While ObjectIterator.AtEnd = False
        Set IxfObject = ObjectIterator.GetObject
        'Read in object
        HandleObject IxfObject, Reader
        'See if it is File object
        Set IxfFile = ReaderHelper.FileReader.CastToFile(IxfObject)
        If Not (IxfFile Is Nothing) Then
            HandleFileObject IxfFile
        End If
        ObjectIterator.Next
    Wend
End Sub

```

Handle a File Object

```

Private Sub HandleFileObject(IxfFile As ISmIxfFile)
    If IxfFile.Embedded = True Then
        IxfFile.Extract App.Path + "\ExtractedFiles"
    End If
End Sub

```

Read an Object

```
Private Sub HandleObject(IxfObject As ISmIxfObject, IxfReader As
SmIxfReader)

    Dim IxfClass As ISmIxfClass
    Dim IxfAttribute As ISmIxfAttribute
    Dim IxfClassBehavior As ISmIxfClassBehavior
    Dim Value As Variant
    Dim Values As ISmIxfAttributesValues
    Dim i, j As Integer

    Set IxfClass =
IxfReader.Schema.Classes.ItemByName(IxfObject.ixfClassName)

    frmSample1.lstObjects.ListItems.Add , , "Object Id = " + IxfObject.Id
+ ":"

    frmSample1.lstObjects.ListItems.Add , , " Class = " + IxfClass.Name

    'class attributes
    If IxfClass.AllAttributes.Count > 0 Then
        frmSample1.lstObjects.ListItems.Add , , " Attributes:"
        Set Values = IxfObject.Values
        For i = 0 To IxfClass.AllAttributes.Count - 1
            Set IxfAttribute = IxfClass.AllAttributes.Item(i)
            Set Value = Values.Item(IxfAttribute.Name)
            If Not VarType(Value) = vbNull Then
                frmSample1.lstObjects.ListItems.Add , , " " +
IxfAttribute.Name + " = " + Value
            End If
        Next
    End If

    'Behavior attributes
    If IxfClass.AllBehaviors.Count > 0 Then
        frmSample1.lstObjects.ListItems.Add , , " BehaviorAttributes:"
        For i = 0 To IxfClass.AllBehaviors.Count - 1
            Set IxfClassBehavior = IxfClass.AllBehaviors.Item(i)
            Set Values = IxfObject.GetBehaviorValues(IxfClassBehavior.URI)
            For j = 0 To IxfClassBehavior.Attributes.Count - 1
                Set IxfAttribute = IxfClassBehavior.Attributes.Item(j)
                Value = Values.Item(IxfAttribute.Name)
                If Not VarType(Value) = vbNull Then
                    frmSample1.lstObjects.ListItems.Add , , " " +
IxfAttribute.Name + " = " + CStr(Value)
                End If
            End If
        End If
    End If
End Sub
```

```

        Next
    Next
End If
frmSample1.lstObjects.ListItems.Add , , ""
End Sub

```

Executing the Application

This section shows how to execute the application.

Note: For further information, please refer to the actual sample application in VB (sample1).

```

Private Sub cmdCreateIXF_Click()
    Dim IxfWriter As SmIxfWriter
    Dim IxfStdHelper As SmIxfStdHelper
    Dim IxfFileName As String

    lstObjects.ListItems.Clear

    dlgIxfFile.ShowOpen
    IxfFileName = dlgIxfFile.FileName
    If IxfFileName = vbNullString Then Exit Sub

    Set IxfWriter = CreateObject("SmartIXF1.SmIxfWriter")
    Set IxfStdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")

    If btnCreateSchema.Value = True Then
        IxfWriter.SetSchemaMode mtEmbedded
        SchemaCreation.CreateSchema IxfWriter.Schema, IxfStdHelper
    Else
        IxfWriter.SetSchemaMode mtExternal,
        "http://example.com/messaging", "sample1.xsd", True
    End If

    IxfWriter.CreateIxfArchiveFile IxfFileName
    DataWriting.CreateData IxfWriter, IxfStdHelper
    IxfWriter.CloseIxfArchiveFile

    MsgBox ("Done")
End Sub

Private Sub cmdReadIxf_Click()
    Dim IxfReader As SmIxfReader
    Dim IxfStdHelper As SmIxfStdHelper

```

```
Dim IxfFileName As String

lstObjects.ListItems.Clear

dlgIxfFile.ShowOpen
IxfFileName = dlgIxfFile.FileName
If IxfFileName = vbNullString Then Exit Sub

Set IxfReader = CreateObject("SmartIXF1.SmIxfReader")
Set IxfStdHelper = CreateObject("SmartIXF1.SmIxfStdHelper")

IxfReader.OpenIxfArchiveFile IxfFileName, True
DataReading.ReadData IxfReader, IxfStdHelper
IxfReader.Close

MsgBox ("Done")
End Sub
```