1 Scope

This document describes the architecture of the PAWS Input Output Subsystem server which provides support for user-implementation of COM components giving or receiving information from the PAWS Run Time System (RTS) during any standard I/O operation such as input, error, warning, informational, or when `INPUT` or `OUTPUT` ATLAS instructions are performed.

The text of this document assumes familiarity with the concepts of COM (Component Object Module).

2 Overview

IOSubsystem identifies a specific resource by its case-sensitive name. Either an *IIOResource* or an *IPersistObject* interface is associated to each resource name in order to allow RTS to localize or instantiate the desired resource. From the user's point of view, a resource may be created programmatically before it is required by RTS. In this case, *IOSubsystem* must maintain the life-cycle of its *IIOResource*. The other possibility is for the user to specify a ProgID for a resource name, letting RTS decide when this resource must be created. In the latter case, the ProgID is contained in a COM *PersistObject*, whose *IPersistObject* is delivered to the *IOSubsystem*.

3 Resource Interfaces

Each resource must implement *IIOResource* which contains general methods and properties modeling a basic resource. Both *ITextResource* and *IBinaryResource* inherit *IIOResource.* Although *IOSubsystem* can manage resources that implement *IIOResource* only, it is pointless for a resource not to have Input/Output methods. Every resource should implement at least one additional interface, either *ITextResource* or *IBinaryResource.*

Each resource can be connected through a connection point to a special *TextPublisher* resource. Generally, *TextPublisher* may be seen as a normal resource as it implements both *ITextResource* and *IBinaryResource* except that it has two outgoing interfaces through which it may notify connected resources when a specific event occurs.

3.1 IIOResource Interface

The *IIOResource* interface has methods and properties that perform standard I/O resource operations. This interface does not include any method that performs an input/output standard operation (input, output, read, write). The interface is a base class for *ITextResource* and *IBinaryResource*.

3.1.1 IDL Description

```
[
	object,
	uuid(3F6B2D01-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IIOResource Interface"),
	pointer_default(unique)
]
interface IIOResource : IUnknown
{
	[helpstring("method Open")]
	HRESULT Open([in] BSTR bstrName, [in] long lMode);
	[helpstring("method Close")]
	HRESULT Close();
	[helpstring("method Flush")]
	HRESULT Flush();
	[helpstring("method Abort")]
```

```
        HRESULT Abort();
        [helpstring("method Seek")]
        HRESULT Seek([in] long lOffset, [in] short sOrigin);
        [propget, helpstring("property Name")]
        HRESULT Name([out, retval] BSTR* pVal);
        [propget, helpstring("property Mode")]
        HRESULT Mode([out, retval] long* pVal);
        [propget, helpstring("property Size")]
        HRESULT Size([out, retval] long* pVal);
        [propget, helpstring("property Position")]
        HRESULT Position([out, retval] long* pVal);
        [propget, helpstring("property Eof")]
        HRESULT Eof([out, retval] VARIANT_BOOL* pVal);
        [propget, helpstring("property State")]
        HRESULT State([out, retval] long* pVal);
};
```

3.1.2 Open method

This method opens a resource.

Parameters

Name	Type	Access	Description
bstrName	BSTR	[in]	File name supposed to be open.
lMode	Long	[in]	Bitwise OR among values of *IOResourceMode* enumeration type.

IOResourceMode bit-field constants

Name	Value	Description
IORSRC_MODE_READ	0x01	The resource is open for reading.
IORSRC_MODE_WRITE	0x02	The resource is open for writing.
IORSRC_MODE_CREATE	0x08	Opens an empty resource for writing. If the given resource exists, its contents are destroyed.
IORSRC_MODE_APPEND	0x08	The resource is open in append mode. [For text files] - Opens for writing at the end of the file (appending) without removing the EOF marker before writing new data to the file; creates the file first if it doesn't exist.
IORSRC_MODE_TEXT	0x10	The resource is open in text mode. [For text files] - Open in text (translated) mode. In this mode, CTRL+Z is interpreted as an end-of-file character on input. Carriage return–linefeed combinations are translated into single linefeeds on input, and linefeed characters are translated to carriage return–linefeed combinations on output.
IORSRC_MODE_BINARY	0x20	The resource is open in binary mode. [For binary files] - Open in binary (untranslated) mode; translations involving carriage-return and linefeed characters are suppressed.
IORSRC_MODE_TYPED	0x40	The resource is open in typed mode. Active when binary mode is selected, too. If this flag is set, the Read method from IBinaryResource first gets the type of the value, and then the value. Write method puts the type of the value, and then the value. If the flag is not set, the type information must not be in the file at reading, and is not written.

3.1.3 Close Method

This method closes a resource.

3.1.4 Flush Method

The *Flush* method flushes the data stream of the resource. It fails if it is called before the resource is open.

3.1.5 Abort Method

This method aborts any input/output operation in progress.

3.1.6 Seek Method

The *Seek* method moves the pointer of the data stream to a specified location.

Parameters

Name	Type	Access	Description
lOffset	long	[in]	The number of bytes the pointer is moved from origin.
sOrigin	short	[in]	One of the values of *IOResourceOrigin* enumeration type.

IOResourceOrigin Enumerated Type

Name	Value	Description
IORSRC_SEEK_SET	0	Beginning of file.
IORSRC_SEEK_CUR	1	Current position of file pointer.
IORSRC_SEEK_END	2	End of file.

3.1.7 Name Property

Type	Access	Description
BSTR	Read-Only	The data stream name.

3.1.8 Mode Property

Type	Access	Description
Long	Read-Only	A bitwise OR among values of *IOResourceMode* enumeration type.

3.1.9 Size Property

Type	Access	Description
long	Read-Only	The size of the data stream.

3.1.10 Position Property

Type	Access	Description
long	Read-Only	The current position in the data stream.

3.1.11 Eof Property

Type	Access	Description
VARIANT_BOOL	Read-Only	VARIANT_TRUE if the position is at the end of the data stream; otherwise, VARIANT_FALSE.

3.1.12 State Property

Type	Access	Description
long	Read-Only	It represents the current state of the resource. It is internally set up as a bitwise OR among values of *IOResourceState* enumeration type.

IOResourceState Enumerated Type

Name	Value	Description
IORSRC_STATE_CLOSED	0x00	The resource was closed successfully.
IORSRC_STATE_OPEN	0x01	The resource was open successfully.
IORSRC_STATE_DIRTY	0x02	An output operation was performed successfully.

3.2 ITextResource Interface

The *ITextResource* interface inherits from the *IIOResource* and adds two new methods.

3.2.1 IDL Description

```
[
	object,
	uuid(3F6B2D03-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("ITextResource Interface"),
	pointer_default(unique)
]
interface ITextResource : IIOResource
{
	[helpstring("method Input")]
	HRESULT Input([in] long lType, [in] BSTR sFmt, [out, retval] BSTR* pVal);
	[helpstring("method Output")]
	HRESULT Output([in] BSTR val);
};
```

3.2.2 Input method

The *Input* method reads a string from the data stream.

Parameters

Name	Type	Access	Description
lType	long	[in]	The ATLAS type expected, as a single value of *IOResourceType* enumeration.
sFmt	BSTR	[in]	Formatting information, for digital values only.
pVal	BSTR	[out, retval]	The data to be read.

IOResourceType bit-field constants

Value	Value	Description
IORSRC_TYPE_NOTAVAILABLE	-1	Type unavailable.
IORSRC_TYPE_BOOL	0	Boolean type.
IORSRC_TYPE_INT	1	Integer type.
IORSRC_TYPE_REAL	2	Real (decimal) type.
IORSRC_TYPE_TEXT	4	Text type.
IORSRC_TYPE_CNX	6	Connection (89) type.
IORSRC_TYPE_DIG	7	Digital type.
IORSRC_TYPE_CON	9	Connection (85) type.

Special Case:

This method gets the string entered by the user in the edit control if the input type is not an IORSRC_TYPE_BOOL. Otherwise, the user has to click either the TRUE or FALSE button. When used in combination with the *Default* property (passing a valid string) and the *UseHistory* property (passing VARIANT_TRUE) of the *IStdIO* interface, an input dialog will no longer be displayed, just the *Default* string is returned.

When the input type is IORSRC_TYPE_DIG (digital type), the second argument of this method with the *Prompt* property of *IStdIO* controls the specific message that is displayed as a prompt in the dialog box. The possible values for this argument are:

Value	Description
"Z"	The input string has to represent a digital binary value.
"O"	The input string has to represent a digital octal value.
"A"	The input string has to represent a digital hexadecimal value.

3.2.3 Output method

The *Output* method writes a string to the file.

Parameters

Name	Type	Access	Description
Val	BSTR	[in]	String that is written in the file.

3.3 IBinaryResource Interface

The *IBinaryResource* interface inherits from the *IIOResource* and adds two new methods.

3.3.1 IDL Description

```
[
	object,
	uuid(3F6B2D02-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IBinaryResource Interface"),
	pointer_default(unique)
]
interface IBinaryResource : IIOResource
{
	[helpstring("method Read")]
	HRESULT Read([in] long lType, [out, retval] VARIANT* pVal);
	[helpstring("method Write")]
	HRESULT Write([in] long lType, [in] VARIANT val);
};
```

3.3.2 Read method

This method reads a standard ATLAS value from the file. If the open mode for the resource is IORSRC_MODE_TYPED, the type of the value must be read from the stream before the value is read; in all other cases only the value will be there for reading. The ATLAS type expected for reading is given by the first argument.

Parameters

Name	Type	Access	Description
lType	Long	[in]	The ATLAS type expected, as a single value of *IOResourceType* enumeration.
pVal	VARIANT*	[out, retval]	The storage for the read value.

3.3.3 Write method

This method writes a standard ATLAS value to the file. If the open mode for the resource is IORSRC_MODE_TYPED, the type of the value is written to the file first, then the value is written; in all other cases only the value is written. The ATLAS type is given by the first argument and the value by the second argument.

Parameters

Name	Type	Access	Description
lType	long	[in]	The ATLAS type, as a single value of IOResourceType enumeration.
Val	VARIANT	[in]	The storage of the written value.

3.4 IStructuredResource Interface

The *IStructuredResource* interface inherits from the *IBinaryResource* overloading both its methods.

3.4.1 IDL Description

```
[
      object,
      uuid(3F6B2D06-F0DA-11D2-BBB0-00C0268914D3),
      oleautomation,
      helpstring("IStructuredResource Interface"),
      pointer_default(unique)
]
interface IStructuredResource : IIOResource
{
      [helpstring("method Read")]
      HRESULT Read([in] VARIANT varTypes, [out, retval] VARIANT* pVal);
      [helpstring("method Write")]
      HRESULT Write([in] VARIANT varTypes, [in] VARIANT val);
};
```

3.4.2 Read method

This method reads a standard ATLAS array or record from the file. If the open mode for the resource is IORSRC_MODE_TYPED, the type of each element must be read from the stream before the value is read; in all other cases only the value will be there for reading. The ATLAS types expected for reading are given by the first argument.

Parameters

Name	Type	Access	Description
varTypes	VARIANT	[in]	A variant of type VT_I4 | VT_ARRAY (Safearray of longs). It defines the type of each element as a long value. The types are similar with those defined at IBinaryResource::Write method.
pVal	VARIANT*	[out, retval]	The pointer's VARIANT type is VT_VARIANT | VT_ARRAY (Safearray of VARIANT). Each element of the array/record is read to a VARIANT in this Safearray. The read elements types depend on the corresponding elements types from *varTypes*. There is a one to one map between elements of *varTypes* and the expected VARIANT from the Safearray.

3.4.3 Write method

This method writes a standard ATLAS array or record to the file. If the open mode for the resource is IORSRC_MODE_TYPED, the type of each element value is written to the file first, then the value is written; in all other cases only the value is written. The ATLAS elements types are given by the first argument and the values by the second argument.

Parameters

Name	Type	Access	Description
varTypes	Long	[in]	A variant of type VT_I4 | VT_ARRAY (Safearray of longs). It defines the type of each element as a long value. These are similar with those defined at IBinaryResource::Read method.
val	VARIANT	[in]	The argument type is VT_ARRAY | VT_VARIANT (Safearray of VARIANT). Each array or record element is a VARIANT in this Safearray. The type of array/record element is kept in its corresponding long from *varTypes*.

3.5 IIOSubsystem Interface

The *IIOSubsystem* interface is the default interface of the *IOSubsystem* object. It exposes the *DataContainer* from *IOSubsystem* as well as methods which can find or create a specific resource.

3.5.1 IDL Description

```
[
       object,
       uuid(3F6B2D11-F0DA-11D2-BBB0-00C0268914D3),
       oleautomation,
       helpstring("IIOSubsystem Interface"),
       pointer_default(unique)
]
interface IIOSubsystem : IUnknown
{
       [helpstring("method Build")]
       HRESULT Build([in] BSTR sName, [out, retval] IIOResource** ppRsrc);
       [helpstring("method Find")]
       HRESULT Find([in] BSTR sName, [out, retval] IIOResource** ppRsrc);
       [propget, helpstring("property ResourceMap")]
       HRESULT ResourceMap([out, retval] IDispatch** pVal);
       [propget, helpstring("property DefaultResource")]
       HRESULT DefaultResource([out, retval] VARIANT* pVal);
       [propput, helpstring("property DefaultResource")]
       HRESULT DefaultResource([in] VARIANT newVal);
};
```

3.5.2 Build method

The *Build* method looks for a resource with the name specified by the first argument. If there is an entry with this name in the *DataContainer*, it must be either a VT_DIPATCH or a VT_UNKNOWN variant. If the object pointed to by variant implements an *IIOResource*, *Build* returns the object. Otherwise the interface must implement an *IPersistObject.* In this case, the resource is taken or created by *Build* method from *IPersistObject*. If there is no entry in the *DataContainer*, a *FileResource* is created. After building the resource, *DataContainer* contains an active entry for the specified name.

Parameters

Name	Type	Access	Description
name	BSTR	[in]	Resource name, as a not-empty case sensitive string.
ppRsrc	IIOResource**	[out, retval]	Address of pointer that receives the interface.

3.5.3 Find Method

The *Find* method looks for the resource with the name specified by the first argument. Unlike *Build* method, *Find* does not create the resource if the name is not found in the *DataContainer*.

Parameters

Name	Type	Access	Description
name	BSTR	[in]	Resource name, as a not-empty case sensitive string.
ppRsrc	IIOResource**	[out, retval]	Address of pointer that receives the interface.

3.5.4 ResourceMap Property.

Type	Access	Description
IDispatch*	Read-Only	Pointer to the *IDispatch* interface of the *IOSubsystem DataContainer* object. The *DataContainer* object implements both *IDataBag* and *IDataColl*.

3.5.5 DefaultResource Property

Type	Access	Description
VARIANT	Read-Only	*IDispatch* interface of the default resource of the *IOSubsystem*.

3.6 ISoundResource Interface

All sound files are played using the PlaySound function from Platform SDK (Software Devolpment Kit), Microsoft's multimedia package. This interface manages the option given to this function as the second argument.

3.6.1 IDL Description

```
[
	object,
	uuid(3F6B2D61-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("ISoundResource Interface"),
	pointer_default(unique)
]
interface ISoundResource : IDispatch
{
	[propget, id(1), helpstring("property Options")]
	HRESULT Options([out, retval] long* pVal);
	[propput, id(1), helpstring("property Options")]
	HRESULT Options([in] long newVal);
};
```

3.6.2 Options Property

Type	Access	Description
long	Read-Write	A bitwise OR of the flags shown below.

Sound Options Values

Value	Description
SND_ALIAS	The string argument of *Output* method is a system-event alias in the registry or the WIN.INI file. This flag must not be used with SND_FILENAME.
SND_FILENAME	The string argument of the *Output* method is a file name.
SND_ASYNC	The sound is played asynchronously and the *Flush* or *Close* method returns immediately after beginning the sound.
SND_NOWAIT	If the driver is busy, the *Flush* or *Close* method returns immediately without playing the sound.
SND_NOSTOP	The specified sound event generated by the *Flush* or *Close* method will yield to another sound event that is already playing. If a sound cannot be played because the resource needed to generate that sound is busy playing another sound, these methods immediately returns E_FAIL without playing the requested sound. If this flag is not specified, *Flush* or *Close* attempts to stop the currently playing sound so that the device can be used to play the new sound.
SND_NODEFAULT	No default sound event is used. If the sound cannot be found, *Flush* or *Close* returns E_FAIL silently without playing the default sound.

3.7 IPrinterResource Interface

This interface controls the printing options.

3.7.1 IDL Description

```
[
	object,
	uuid(3F6B2D41-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IPrinterResource Interface"),
	pointer_default(unique)
]
interface IPrinterResource : IDispatch
{
	[propget, id(1), helpstring("property CmdLine")]
	HRESULT CmdLine([out, retval] BSTR* pVal);
	[propput, id(1), helpstring("property CmdLine")]
	HRESULT CmdLine([in] BSTR newVal);
	[propget, id(2), helpstring("property TimeOut")]
	HRESULT TimeOut([out, retval] long* pVal);
	[propput, id(2), helpstring("property TimeOut")]
	HRESULT TimeOut([in] long newVal);
	[propget, id(3), helpstring("property Port")]
	HRESULT Port([out, retval] BSTR* pVal);
	[propput, id(3), helpstring("property Port")]
	HRESULT Port([in] BSTR newVal);
	[propget, id(4), helpstring("property PageLength")]
	HRESULT PageLength([out, retval] short* pVal);
	[propput, id(4), helpstring("property PageLength")]
	HRESULT PageLength([in] short newVal);
	[propget, id(5), helpstring("property File")]
	HRESULT File([out, retval] BSTR* pVal);
	[propput, id(5), helpstring("property File")]
	HRESULT File([in] BSTR newVal);
	[propget, id(6), helpstring("property Type")]
	HRESULT Type([out, retval] PrinterType* pVal);
	[propput, id(6), helpstring("property Type")]
	HRESULT Type([in] PrinterType newVal);
};
```

3.7.2 CmdLine Property

Type	**Access**	**Description**
BSTR	Read-Write	The DOS command line with which printing is accomplished, assuming the printer type is PRINTER_TYPE_WINDOWS.

3.7.3 TimeOut Property

Type	**Access**	**Description**
Long	Read-Write	It represents the timeout for printing in milliseconds, if the printer type is PRINTER_TYPE_WINDOWS. The default value for this property is –1, which means no timeout.

3.7.4 Port Property

Type	**Access**	**Description**
BSTR	Read-Write	It represents the port name where printing is accomplished assuming the printer type is PRINTER_TYPE_PORT. The default value for this property is ”LPT1”.

3.7.5 PageLength Property

Type	Access	Description
Short	Read-Write	It represents the printer page length, if the printer type is PRINTER_TYPE_PORT. The default value for this property is zero indicating that the page length is infinite.

3.7.6 File Property

Type	Access	Description
BSTR	Read-Write	It represents the name of file where the data will be written, if the printer type is PRINTER_TYPE_FILE. The default property value is **PRINTER**.

3.7.7 Type Property

Type	Access	Description
PrinterType	Read-Write	It represents the type of printer as one of the enumeration values shown below. The default value is PRINTER_TYPE_WINDOWS.

***PrinterType* values**

Name	Value	Description
PRINTER_TYPE_WINDOWS	0	Windows type printer: the printing is made using the Windows driver.
PRINTER_TYPE_PORT	1	Port type printer: the printing is made to a port.
PRINTER_TYPE_FILE	2	File type printer: the printing is made in a file.

3.8 IStdIO Interface

This interface controls/prompts user for input.

3.8.1 IDL Description

```
[
	object,
	uuid(3F6B2D04-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IStdIO Interface"),
	pointer_default(unique)
]
interface IStdIO : ITextResource
{
	[propget, helpstring("property Prompt")]
	HRESULT Prompt([out, retval] BSTR* pVal);
	[propput, helpstring("property Prompt")]
	HRESULT Prompt([in] BSTR newVal);
	[propget, helpstring("property Default")]
	HRESULT Default([out, retval] BSTR* pVal);
	[propput, helpstring("property Default")]
	HRESULT Default([in] BSTR newVal);
	[propget, helpstring("property UseHistory")]
	HRESULT UseHistory([out, retval] VARIANT_BOOL* pVal);
	[propput, helpstring("property UseHistory")]
	HRESULT UseHistory([in] VARIANT_BOOL newVal);
};
```

3.8.2 Prompt Property

If this property is not NULL, its string is shown as a text of a static control in the dialog box where the edit control or the TRUE/FALSE buttons are. If the string is empty, a default message is put in the dialog instead, depending on the required mode (first argument of *Input* method). The default value for this property is NULL.

Type	Access	Description
BSTR	Read-Write	If not empty, this string will be displayed in the input dialog box.

3.8.3 Default Property

If this property is not NULL, and the *UseHistory* property is a VARIANT_TRUE, the property string is returned by the *Input* method, and no dialog box is opened. The default value for this property is NULL.

Type	Access	Description
BSTR	Read-Write	The returned value by *Input* if the *UseHistory* is VARIANT_TRUE.

3.8.4 UseHistory Property

This property controls the returned value of *Input* method. See *Default* property above. The default value for this property is VARIANT_FALSE.

Type	Access	Description
VARIANT_BOOL	Read-Write	It controls the returned value of *Input* method.

3.9 IDlgInputResource interface

This interface controls the Windows elements of which the input dialog box must be aware.
PosX and *PosY* are persistent properties and are loaded or saved by the specific methods of *IPersistStreamInit* or *IPersistPropertyBag*.

3.9.1 IDL Description

```
[
	object,
	uuid(3F6B2D81-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IDlgInputResource Interface"),
	pointer_default(unique)
]
interface IDlgInputResource : IDispatch
{
	[propget, id(1), helpstring("property ParentWnd")]
	HRESULT ParentWnd([out, retval] long *pVal);
	[propput, id(1), helpstring("property ParentWnd")]
	HRESULT ParentWnd([in] long newVal);
	[propget, id(2), helpstring("property PosX")]
	HRESULT PosX([out, retval] ULONG *pVal);
	[propput, id(2), helpstring("property PosX")]
	HRESULT PosX([in] ULONG newVal);
	[propget, id(3), helpstring("property PosY")]
	HRESULT PosY([out, retval] ULONG *pVal);
	[propput, id(3), helpstring("property PosY")]
	HRESULT PosY([in] ULONG newVal);
};
```

3.9.2 ParentWnd Property

Type	Access	Description
Long	Read-Write	Represents the parent window handle for the dialog box that *Input* method may create. The default value for this property is NULL.

3.9.3 PosX Property

Type	Access	Description
ULONG	Read-Write	Represents the X coordinate for the dialog box, in pixels, from the left margin of the parent window. If the *ParentWnd* property is NULL, distance is measured from the left margin of the screen. The default value for this property is zero.

3.9.4 PosY Property

Type	Access	Description
ULONG	Read-Write	Represents the Y coordinate for the dialog box, in pixels, from the top margin of the parent window. If the *ParentWnd* property is NULL, distance is measured from the top margin of the screen. The default value for this property is zero.

3.10 IIOControl Interface

3.10.1 IDL Description

```
[
	object,
	uuid(3F6B2D05-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IIOControl Interface"),
	pointer_default(unique)
]
interface IIOControl : IUnknown
{
	[helpstring("method Commit")]
	HRESULT Commit();
	[helpstring("method OpenEx")]
	HRESULT OpenEx([in] BSTR bstrName,
	                [in] long lMode,
	                [in] VARIANT varTypes,
	                [in] VARIANT val);
	[helpstring("method SeekInLine")]
	HRESULT SeekInLine([in] long lOffset, [in] short shOrigin);
};
```

3.10.2 Commit method

Commits changes made to the object. This method is called by the RTS when the TPS finishes.

3.10.3 OpenEx method

Parameters

Name	Type	Access	Description
bstrName	BSTR	[in]	Name of the file to be opened.
lMode	long	[in]	Mode in which file is opened. Must be a value from the *IOResourceMode* enumeration.
varTypes	VARIANT	[in]	Reserved data used by RTS
Val	VARIANT	[in]	Reserved data used by RTS

3.10.4 SeekInLine method

Parameters

Name	Type	Access	Description
lOffset	long	[in]	Offset.
shOrigin	short	[in]	Position from which offset is calculated.One of the values of *IOResourceOrigin* enumeration type

3.11 IMiscellaneous Interface

3.11.1 IDL Description

```
[
	object,
	uuid(3F6B2D09-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IMiscellaneous Interface"),
	pointer_default(unique)
]
interface IMiscellaneous : IUnknown
{
	[propget, helpstring("property FileHandle")]
	HRESULT FileHandle([out, retval] long * pVal);
};
```

3.11.2 FileHandle Property

Type	Access	Description
Long	Read-Only	Provides the file handle.

3.12 IObjectFactory Interface

This interface controls object creation given a PROGID.

3.12.1 IDL Description

```
[
	object,
	uuid(3F6B2D12-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("IObjectFactory Interface"),
	pointer_default(unique)
]
interface IObjectFactory : IUnknown
{
	[helpstring("method CreateObject")]
	HRESULT CreateObject([in] BSTR sProgID, [out, retval] IUnknown** pVal);
};
```

3.12.2 CreateObject method

This method allows a user to creates an Object given a ProgID.

Parameters

Name	Type	Access	Description
sProgID	BSTR	[in]	ProgID of the object to be created
pVal	IUnknown**	[out, retval]	Pointer to the Created object

3.13 IFileResource Interface

3.13.1 IDL Description

```
[
       object,
       uuid(3F6B2D31-F0DA-11D2-BBB0-00C0268914D3),
       dual,
       helpstring("IFileResource Interface"),
       pointer_default(unique)
]
interface IFileResource : IDispatch
{
       [propget, id(1), helpstring("property TextInputMode")]
       HRESULT TextInputMode([out, retval] TextFileInputMode* pVal);
       [propput, id(1), helpstring("property TextInputMode")]
       HRESULT TextInputMode([in] TextFileInputMode newVal);
};
```

3.13.2 TextInputMode Property

Type	Access	Description
TextFileInputMode	Read-Write	Takes one of the values from the TextFileInputMode enumeration.

TextFileInputMode enumeration constants

Value	Value	Description
TXT_INPUT_IEEE716_89	0	
TXT_INPUT_WORD_SEP	1	

3.14 IPortResource Interface

3.14.1 IDL Description

```
[
       object,
       uuid(3F6B2D51-F0DA-11D2-BBB0-00C0268914D3),
       dual,
       helpstring("IPortResource Interface"),
       pointer_default(unique)
]
interface IPortResource : IDispatch
{
       [propget, id(1), helpstring("property Port")]
       HRESULT Port([out, retval] BSTR* pVal);
       [propput, id(1), helpstring("property Port")]
       HRESULT Port([in] BSTR newVal);
       [propget, id(2), helpstring("property PageLength")]
       HRESULT PageLength([out, retval] short* pVal);
       [propput, id(2), helpstring("property PageLength")]
       HRESULT PageLength([in] short newVal);
};
```

3.14.2 Port Property

Type	Access	Description
BSTR	Read-Write	Port address as a string.

3.14.3 PageLength Property

Type	Access	Description
short	Read-Write	Length of the page.

3.15 IHtmlResource Interface

This interface controls display of an HTML Input GUI for various ATLAS actions and instructions.

3.15.1 IDL Description

```
[
	object,
	uuid(3F6B2D71-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IHtmlResource Interface"),
	pointer_default(unique)
]
interface IHtmlResource : IDispatch
{
	[id(1), helpstring("method Display")]
	HRESULT Display([in] BSTR url);
	[propget, id(2), helpstring("property PosX")]
	HRESULT PosX([out, retval] long* pVal);
	[propput, id(2), helpstring("property PosX")]
	HRESULT PosX([in] long newVal);
	[propget, id(3), helpstring("property PosY")]
	HRESULT PosY([out, retval] long* pVal);
	[propput, id(3), helpstring("property PosY")]
	HRESULT PosY([in] long newVal);
	[propget, id(4), helpstring("property Width")]
	HRESULT Width([out, retval] long* pVal);
	[propput, id(4), helpstring("property Width")]
	HRESULT Width([in] long newVal);
	[propget, id(5), helpstring("property Height")]
	HRESULT Height([out, retval] long* pVal);
	[propput, id(5), helpstring("property Height")]
	HRESULT Height([in] long newVal);
	[propget, id(6), helpstring("property WindowState")]
	HRESULT WindowState([out, retval] long* pVal);
	[propput, id(6), helpstring("property WindowState")]
	HRESULT WindowState([in] long newVal);
};
```

3.15.2 Display method

Displays a given URL as an html page.

Parameters

Name	Type	Access	Description
url	BSTR	[in]	URL of the page to be displayed.

3.15.3 PosX Property

Type	Access	Description
Long	Read-Write	X position of the top left corner of the page to be displayed.

3.15.4 PosY Property

Type	Access	Description
Long	Read-Write	Y position of the top left corner of the page to be displayed.

3.15.5 Width Property

Type	Access	Description
Long	Read-Write	Width of the page to be displayed.

3.15.6 Height Property

Type	Access	Description
Long	Read-Write	Height of the page to be displayed.

3.15.7 WindowState Property

Type	Access	Description
Long	Read-Write	WindowState of the page to be displayed Must be a value from the *IOWindowState* enumeration.

IOWindowState enumeration constants

Value	Value	Description
IORSRC_WINDOW_NORMAL	0	Shows the window normally.
IORSRC_WINDOW_MINIMIZED	1	Shows the window minimized.
IORSRC_WINDOW_MAXIMIZED	2	Shows the window maximized.

3.16 IVideoResource Interface

This interface controls the play and display of video files.

3.16.1 IDL Description

```
[
	object,
	uuid(3F6B2D91-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IVideoResource Interface"),
	pointer_default(unique)
]
interface IVideoResource : IDispatch
{
	[id(1), helpstring("method PlayMovie")]
	HRESULT PlayMovie([in] BSTR sPath);
	[propget, id(2), helpstring("property PosX")]
	HRESULT PosX([out, retval] long* pVal);
	[propput, id(2), helpstring("property PosX")]
	HRESULT PosX([in] long newVal);
	[propget, id(3), helpstring("property PosY")]
	HRESULT PosY([out, retval] long* pVal);
	[propput, id(3), helpstring("property PosY")]
	HRESULT PosY([in] long newVal);
};
```

3.16.2 PlayMovie method

This method allows a user to play a movie file at the position specified.

Parameters

Name	Type	Access	Description
sPath	BSTR	[in]	Path to a file containing the movie file to be played.

3.16.3 PosX Property

Type	Access	Description
Long	Read-Write	X position of the top left corner of the movie window.

3.16.4 PosY Property

Type	Access	Description
Long	Read-Write	Y position of the top left corner of the movie window.

3.17 ICfgFileResource Interface

3.17.1 IDL Description

```
[
	object,
	uuid(3F6B2DA1-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("ICfgFileResource Interface"),
	pointer_default(unique)
]
interface ICfgFileResource : IDispatch
{
	[propput, id(1), helpstring("property Name")]
	HRESULT Name([in] BSTR newVal);
	[propget, id(1), helpstring("property Name")]
	HRESULT Name([out, retval] BSTR* pVal);
	[propput, id(2), helpstring("property Mode")]
	HRESULT Mode([in] long newVal);
	[propget, id(2), helpstring("property Mode")]
	HRESULT Mode([out, retval] long* pVal);
};
```

3.17.2 Name Property

Type	Access	Description
BSTR	Read-Write	Name of the configuration file.

3.17.3 Mode Property

Type	Access	Description
Long	Read-Write	Mode in which the configuration file is opened. Must correspond to one of the values in the *IOResourceMode* enumeration.

3.18 IVideoResourceControl Interface

3.18.1 IDL Description

```
[
	object,
	uuid(3F6B2DD1-F0DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IVideoResourceControl Interface"),
	pointer_default(unique)
]
interface IVideoResourceControl : IDispatch
{
	[id(1), helpstring("method RenderFile")]
	HRESULT RenderFile([in] BSTR val);
	[propget, id(2), helpstring("property PosX")]
	HRESULT PosX([out, retval] long* pVal);
	[propget, id(3), helpstring("property PosY")]
	HRESULT PosY([out, retval] long* pVal);
	[propget, id(4), helpstring("property Width")]
	HRESULT Width([out, retval] long* pVal);
	[propget, id(5), helpstring("property Height")]
	HRESULT Height([out, retval] long* pVal);
	[id(6), helpstring("method Run")]
	HRESULT Run();
	[id(7), helpstring("method GetEventHandle")]
	HRESULT GetEventHandle(long* pVal);
	[id(8), helpstring("method GetEvent")]
	HRESULT GetEvent(long* pVal);
};
```

3.18.2 RenderFile method

This method is used to render a movie file.

Parameters

Name	Type	Access	Description
val	BSTR	[in]	Path to a file containing the movie file to be played.

3.18.3 PosX Property

Type	Access	Description
long	Read-Only	X position of the top left corner of the movie window.

3.18.4 PosY Property

Type	Access	Description
Long	Read-Only	Y position of the top left corner of the movie window.

3.18.5 Width Property

Type	Access	Description
Long	Read-Only	Width of the movie file window to be displayed.

3.18.6 Height Property

Type	Access	Description
Long	Read-Only	Height of the movie file window to be displayed.

3.18.7 Run method

This method is used to play a movie file.

3.18.8 GetEventHandle method

Parameters

Name	Type	Access	Description
pVal	Long	[in]	Returns the handle to the event.

3.18.9 GetEvent method

Parameters

Name	Type	Access	Description
pVal	Long	[in]	Returns an event.

3.19 IRS232Resource Interface

3.19.1 IDL Description

```
[
	object,
	uuid(3F6B2D11-00DA-11D2-BBB0-00C0268914D3),
	dual,
	helpstring("IRS232Resource Interface"),
	pointer_default(unique)
]
interface IRS232Resource : IDispatch
{
	[propget, id(1), helpstring("property Port")]
	HRESULT Port([out, retval] BSTR *pVal);
	[propput, id(1), helpstring("property Port")]
	HRESULT Port([in] BSTR newVal);
	[propget, id(2), helpstring("property Baudrate")]
	HRESULT Baudrate([out, retval] long *pVal);
	[propput, id(2), helpstring("property Baudrate")]
	HRESULT Baudrate([in] long newVal);
	[propget, id(3), helpstring("property CharSize")]
	HRESULT CharSize([out, retval] long *pVal);
	[propput, id(3), helpstring("property CharSize")]
	HRESULT CharSize([in] long newVal);
	[propget, id(4), helpstring("property StopBits")]
	HRESULT StopBits([out, retval] double *pVal);
	[propput, id(4), helpstring("property StopBits")]
	HRESULT StopBits([in] double newVal);
	[propget, id(5), helpstring("property Timeout")]
	HRESULT Timeout([out, retval] long *pVal);
	[propput, id(5), helpstring("property Timeout")]
	HRESULT Timeout([in] long newVal);
	[propget, id(6), helpstring("property NullCharDiscarded")]
	HRESULT NullCharDiscarded([out, retval] VARIANT_BOOL *pVal);
	[propput, id(6), helpstring("property NullCharDiscarded")]
	HRESULT NullCharDiscarded([in] VARIANT_BOOL newVal);
	[propget, id(7), helpstring("property DSRSensitivity")]
	HRESULT DSRSensitivity([out, retval] VARIANT_BOOL *pVal);
	[propput, id(7), helpstring("property DSRSensitivity")]
	HRESULT DSRSensitivity([in] VARIANT_BOOL newVal);
	[propget, id(8), helpstring("property TermChar")]
	HRESULT TermChar([out, retval] long *pVal);
	[propput, id(8), helpstring("property TermChar")]
	HRESULT TermChar([in] long newVal);
```

```
        [propget, id(9), helpstring("property Parity")]
        HRESULT Parity([out, retval] RS232Parity *pVal);
        [propput, id(9), helpstring("property Parity")]
        HRESULT Parity([in] RS232Parity newVal);
        [propget, id(10), helpstring("property ErrorReplacement")]
        HRESULT ErrorReplacement([out, retval] VARIANT_BOOL *pVal);
        [propput, id(10), helpstring("property ErrorReplacement")]
        HRESULT ErrorReplacement([in] VARIANT_BOOL newVal);
        [propget, id(11), helpstring("property ReplacementChar")]
        HRESULT ReplacementChar([out, retval] long *pVal);
        [propput, id(11), helpstring("property ReplacementChar")]
        HRESULT ReplacementChar([in] long newVal);
        [propget, id(12), helpstring("property FlowControl")]
        HRESULT FlowControl([out, retval] RS232FlowControl* pVal);
        [propput, id(12), helpstring("property FlowControl")]
        HRESULT FlowControl([in] RS232FlowControl newVal);
        [propget, id(13), helpstring("property TxContinueOnXoff")]
        HRESULT TxContinueOnXoff([out, retval] VARIANT_BOOL *pVal);
        [propput, id(13), helpstring("property TxContinueOnXoff")]
        HRESULT TxContinueOnXoff([in] VARIANT_BOOL newVal);
        [propget, id(14), helpstring("property XonChar")]
        HRESULT XonChar([out, retval] long *pVal);
        [propput, id(14), helpstring("property XonChar")]
        HRESULT XonChar([in] long newVal);
        [propget, id(15), helpstring("property XoffChar")]
        HRESULT XoffChar([out, retval] long *pVal);
        [propput, id(15), helpstring("property XoffChar")]
        HRESULT XoffChar([in] long newVal);
        [id(16), helpstring("method ApplySettings")]
        HRESULT ApplySettings();
};
```

3.19.2 Port Property

Type	Access	Description
BSTR	Read-Write	Communication port as a string.

3.19.3 BaudRate Property

Type	Access	Description
Long	Read-Write	Communication baud rate.

3.19.4 CharSize Property

Type	Access	Description
Long	Read-Write	Size of character in bits.

3.19.5 StopBits Property

Type	Access	Description
Double	Read-Write	Number of stop bits.

3.19.6 Timeout Property

Type	Access	Description
Long	Read-Write	Communication timeout value.

3.19.7 NullCharDiscarded Property

Type	Access	Description
VARIANT_BOOL	Read-Write	Controls if NULL character discarded.

3.19.8 DSRSensitivity Property

Type	Access	Description
VARIANT_BOOL	Read-Write	Controls if the DSR line is active.

3.19.9 TermChar Property

Type	Access	Description
Long	Read-Write	Termination character.

3.19.10Parity Property

Type	Access	Description
RS232Parity	Read-Write	Contains a value from the RS232Parity enumeration.

RS232Parity enumeration constants

Value	Value	Description
RS232PARITY_NONE	0	Parity none.
RS232PARITY_ODD	1	Parity odd.
RS232PARITY_EVEN	2	Parity even.
RS232PARITY_MARK	3	Parity mark.
RS232PARITY_SPACE	4	Parity space.

3.19.11ErrorReplacement Property

Type	Access	Description
VARIANT_BOOL	Read-Write	Controls if error replacement is used.

3.19.12ReplacementChar Property

Type	Access	Description
long	Read-Write	Replacement character for an error character if error replacement is used.

3.19.13FlowControl Property

Type	Access	Description
RS232FlowControl	Read-Write	Contains a value from the RS232FlowControl enumeration.

RS232FlowControl enumeration constants

Value	Value	Description
RS232FLOWCONTROL_NONE	0	Parity none.
RS232FLOWCONTROL_SOFTWARE	1	Parity odd.
RS232FLOWCONTROL_HARDWARE	2	Parity even.

3.19.14TxContinueOnXoff Property

Type	Access	Description
VARIANT_BOOL	Read-Write	Controls if device can continue transmission (send data) if its input buffer is full. Used for software flow control mode only.

3.19.15XonChar Property

Type	Access	Description
Long	Read-Write	Communication character to provide software flow control with sender. Notifies the sender that buffer is empty and it can send data. Used for software flow control mode only.

3.19.16XoffChar Property

Type	Access	Description
Long	Read-Write	Communication character to provide software flow control with sender. Notifies the sender that buffer is full to stop sending this device data. Used for software flow control mode only.

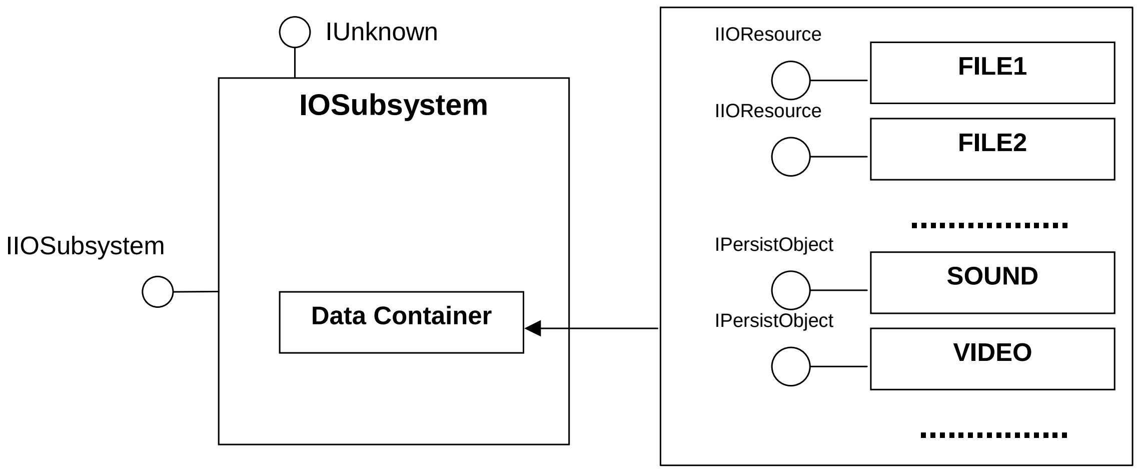
3.19.17ApplySettings method

This method allows the user to commit the current property settings to the communication master.

4 IOSubsystem Component

The Input/Output Subsystem Component is internally created and exposes *IIOSubsystem*. This interface is available to the RTS in order to find, or to create and configure a particular resource. In order to programmatically add or remove a specific resource, the user must use this same interface.

The *IOSubsystem* contains a *DataContainer*. Each element of this container has a case-sensitive name and a VARIANT of type VT_UNKNOWN or VT_DISPATCH. The *IUnknown* or *IDispatch* interface contained in this variant has to be either an *IIOResource* or an *IPersistObject*.

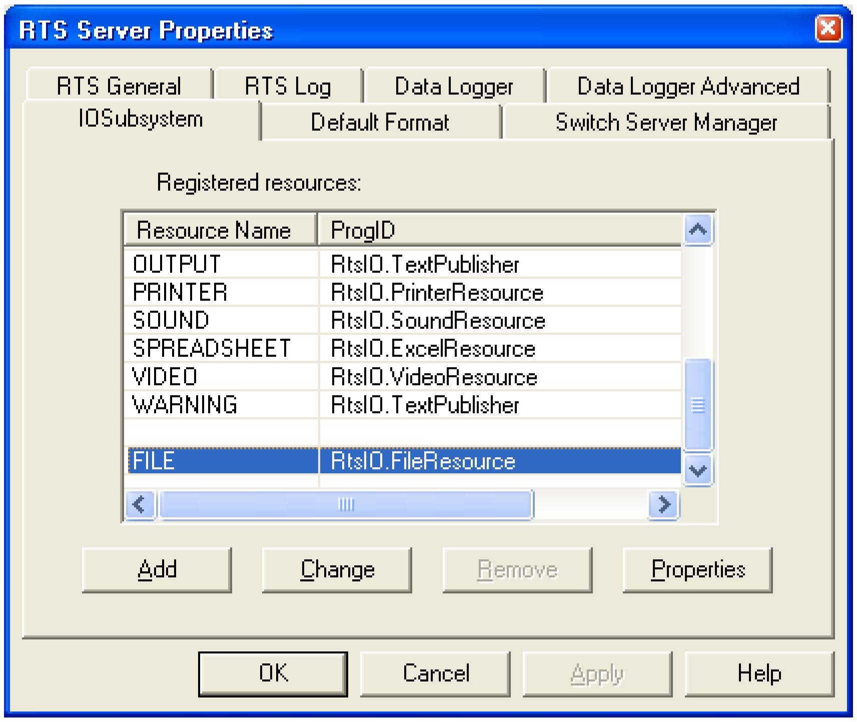


The user may consider two approaches:

First, programmatically creating a specific resource, this must at least implement IIOResource. The resource name and its interface must be added programmatically to the *DataContainer* from *IOSubsystem* as a distinctive container element. Subsequently, each time RTS asks for a resource with this name, this *IIOResource* interface is obtained. No additional resource is created, the initial configuration and the saving of configuration being the user's responsibilities. These tasks must be done programmatically using the *IOSubsystem* property page, even if RTS does allow the user to change the configuration of this resource, by using the *IUnknown* or *IDispatch* from the *DataContainer*'s variant for this resource is an IIOResource interface.

Second, specify a user resource using its *ProgID*. This must be done either manually using the Add button from *IOSubsystem* property page, or by programmatically creating a *PersistObject* which has as a property the user's resource ProgID.

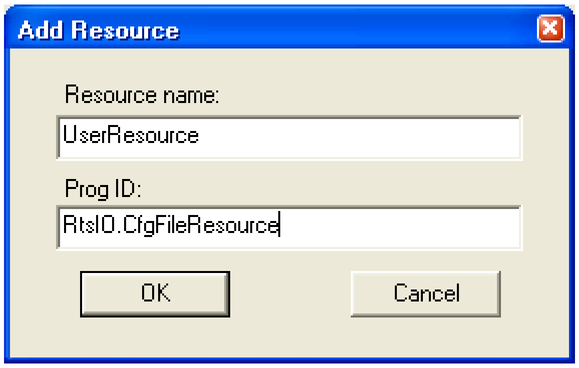
The property page for *IOSubsystem* is shown below:



Registered resources:

Resource Name	ProgID
OUTPUT	RtsIO.TextPublisher
PRINTER	RtsIO.PrinterResource
SOUND	RtsIO.SoundResource
SPREADSHEET	RtsIO.ExcelResource
VIDEO	RtsIO.VideoResource
WARNING	RtsIO.TextPublisher
FILE	RtsIO.FileResource

The **Add** button is always active and lets the user add a name and a ProgID. Duplicate or empty resource names are not allowed. Even if the name of the resource is case-sensitive, it is not recommended for two resources to have the same name. In spite of the capability of the RTS and the *IOSubsystem* to manage this kind of a situation correctly, PAWS Developer Studio considers the names of its file resources as upper case strings.
When the **Add** button is clicked, the following dialog is displayed:



Resource name: UserResource

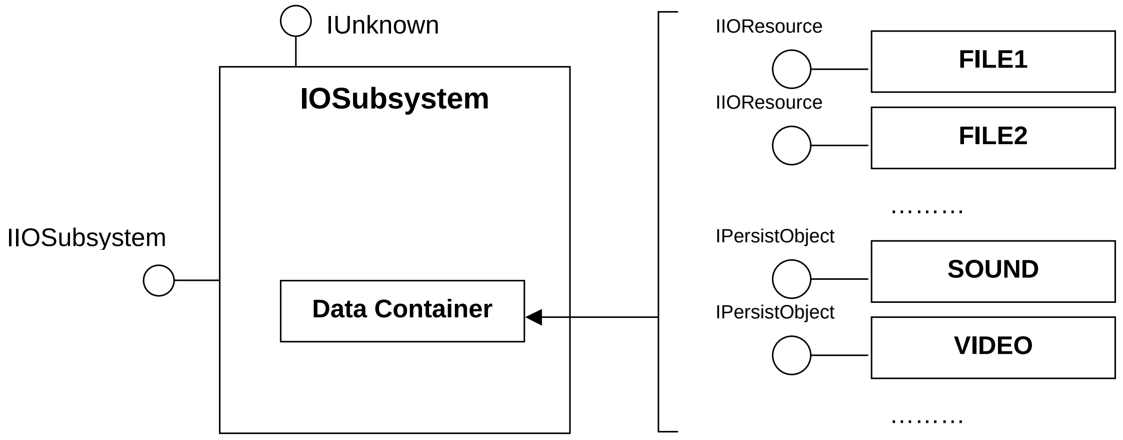
Prog ID: RtsIO.CfgFileResource

The **Remove** button in the IOSubsystem Control Properties dialog is activated every time a resource is selected in the list control. If the selected resource is removed, its configuration is not saved when the IOSubsystem configuration is saved. If the resource is already open, the RTS keeps an internal reference for the old resource and uses it until the resource is closed or the project is unloaded.

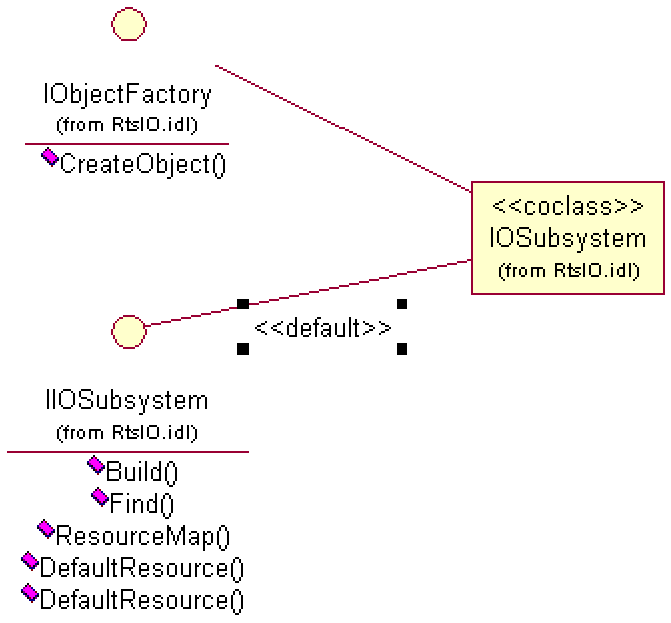
The **Properties** button is active each time the selected resource from list control implements *ISpecifyPropertyPages* if the resource has already been created. If the resource has not been created, the **Properties** button may be active if only the COM object whose ProgID is selected implements both *ISpecifyPropertyPages* and *IPersistStreamInit* interfaces.

Resource configuration is saved only for resources specified using *IPersistObject* interfaces and if the resource object implements the *IPersistStreamInit* interface. Resources added using the *IOSubsystem* property page always use *PersistObject* containers.

As mentioned before, each element from the *IOSubsystem DataContainer* must have as value type a VARIANT of VT_UNKNOWN or VT_DISPATCH type. There is no connection between the type of the variant and the type of the resource kept in the variant. The object whose interface is contained in VARIANT must implement either an IPersistObject or an IIOResource.



4.1 UML Description



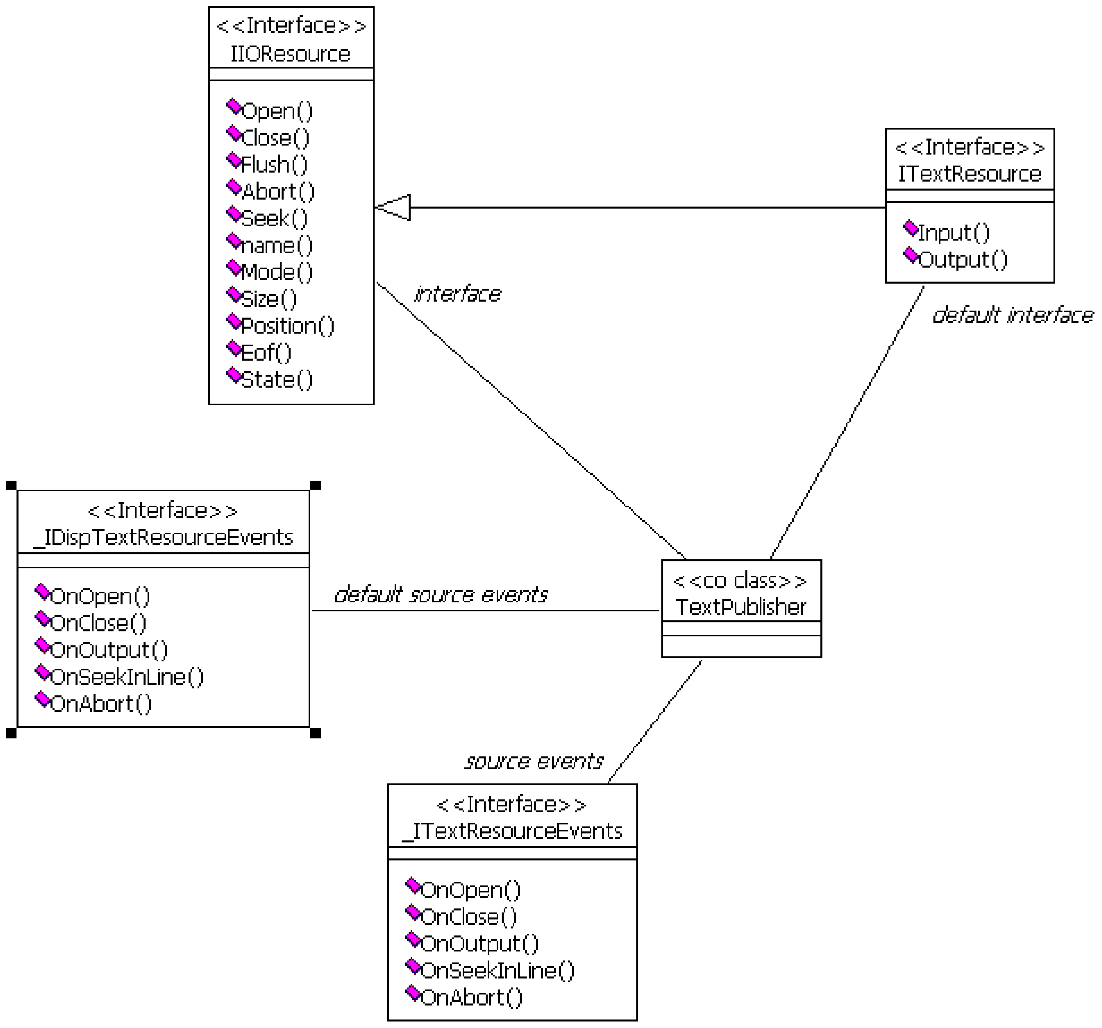
4.2 IDL description

```
[
	uuid(3F6B2D10-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("IOSubsystem Class")
]
coclass IOSubsystem
{
	[default] interface IIOSubsystem;
	interface IObjectFactory;
};
```

5 TextPublisher Component

This component is used for sending notification to the resources that are connected to it, through *_ITextResourceEvents* or *_IDispTextResourceEvents.*

5.1 UML Description



5.2 IDL Description

```
[
	uuid(3F6B2D20-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("TextPublisher Class")
]
coclass TextPublisher
{
	[default] interface ITextResource;
	interface IIOResource;
	[default, source] dispinterface _IDispTextResourceEvents;
	[source] interface _ITextResourceEvents;
};
```

5.3 _ITextResourceEvents Interface

This is an outgoing oleautomation interface for the *TextPublisher*.

5.3.1 IDL Description

```
[
	object,
	uuid(3F6B2D07-F0DA-11D2-BBB0-00C0268914D3),
	oleautomation,
	helpstring("_ITextResourceEvents Interface"),
	pointer_default(unique)
]
interface _ITextResourceEvents : IUnknown
{
	[helpstring("method OnOpen")]
	HRESULT OnOpen([in] BSTR name);
	[helpstring("method OnClose")]
	HRESULT OnClose();
	[helpstring("method OnOutput")]
	HRESULT OnOutput([in] BSTR val);
	[helpstring("method OnSeekInLine")]
	HRESULT OnSeekInLine([in]long lOffset, [in]short shOrigin);
	[helpstring("method OnAbort")]
	HRESULT OnAbort();
};
```

5.3.2 OnOpen

Event generated when *Open* method is called for the resource.

5.3.3 OnClose

Event generated when *Close* method is called for the resource.

5.3.4 OnOutput

Event generated when *Flush* method is called for the resource.

5.3.5 OnSeekInLine

Event generated when *SeekInLine* method is called for the resource.

5.3.6 OnAbort

Event generated when *Abort* method is called for the resource.

5.4 _IDispTextResourceEvents Interface

This is an outgoing dispinterface for the *TextPublisher*.

5.4.1 IDL Description

```
[
	uuid(3F6B2D08-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("_IDispTextResourceEvents Interface")
]
dispinterface _IDispTextResourceEvents
{
	[id(1), helpstring("method OnOpen")]
	HRESULT OnOpen([in] BSTR name);
	[id(2), helpstring("method OnClose")]
	HRESULT OnClose();
	[id(3), helpstring("method OnOutput")]
	HRESULT OnOutput([in] BSTR val);
	[id(4), helpstring("method OnSeekInLine")]
	HRESULT OnSeekInLine([in]long lOffset, [in]short shOrigin);
	[id(5), helpstring("method OnAbort")]
	HRESULT OnAbort();
};
```

5.4.2 OnOpen

Event generated when *Open* method is called for the resource.

5.4.3 OnClose

Event generated when *Close* method is called for the resource.

5.4.4 OnOutput

Event generated when *Flush* method is called for the resource.

5.4.5 OnSeekInLine

Event generated when *SeekInLine* method is called for the resource.

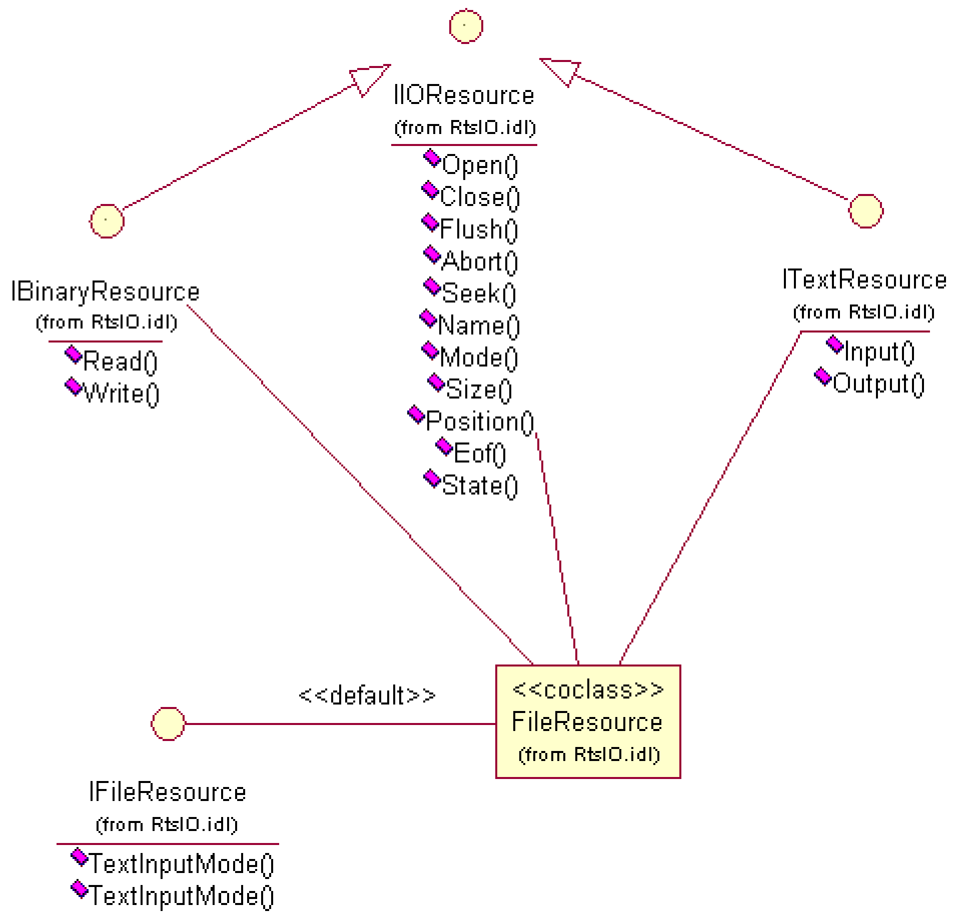
5.4.6 OnAbort

Event generated when *Abort* method is called for the resource.

6 FileResource Component

The *FileResource* component is a standard file resource, and each time the *Build* method from *IIOSubsystem* does not find an entry for the resource name in the *IOSubsystem DataContainer*, this type of resource is created instead.
Other that its default interface *IFileResource*, the *FileResource* component also implements *IBinaryResource* , *ITextResource*. Both these interfaces derive from *IIOResource*.

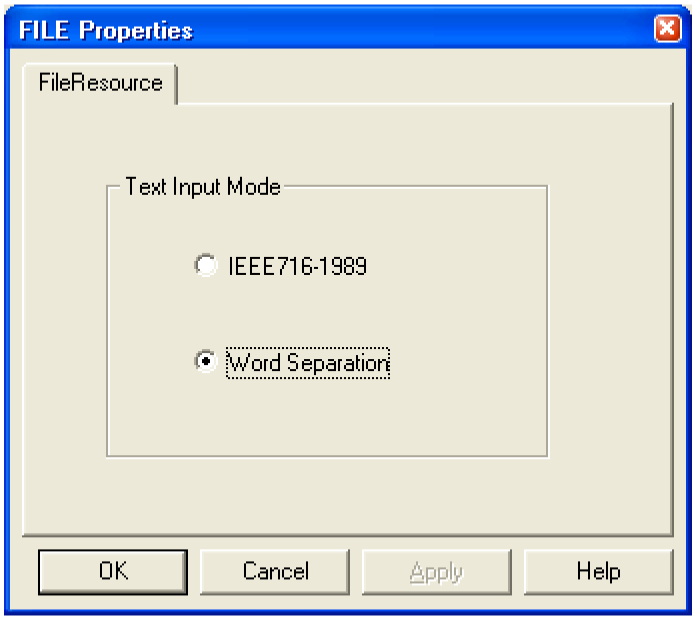
6.1 UML Description



6.2 IDL Description

```
[
	uuid(3F6B2D30-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("FileResource Class")
]
coclass FileResource
{
	[default] interface IFileResource;
	interface IBinaryResource;
	interface IIOResource;
	interface ITextResource;
};
```

6.3 Property Pages


FILE Properties

FileResource

Text Input Mode

( ) IEEE716-1989

[✓ Word Separation]

The display shows how to control the properties by clicking the “Properties” of the “FILE” Resource.

7 PrinterResource Component

There are three options for printing.

♦ ♦ Through the Windows printer driver, using an intermediate program specified on the DOS command line. The default command line for printing uses the “notepad.exe” program.

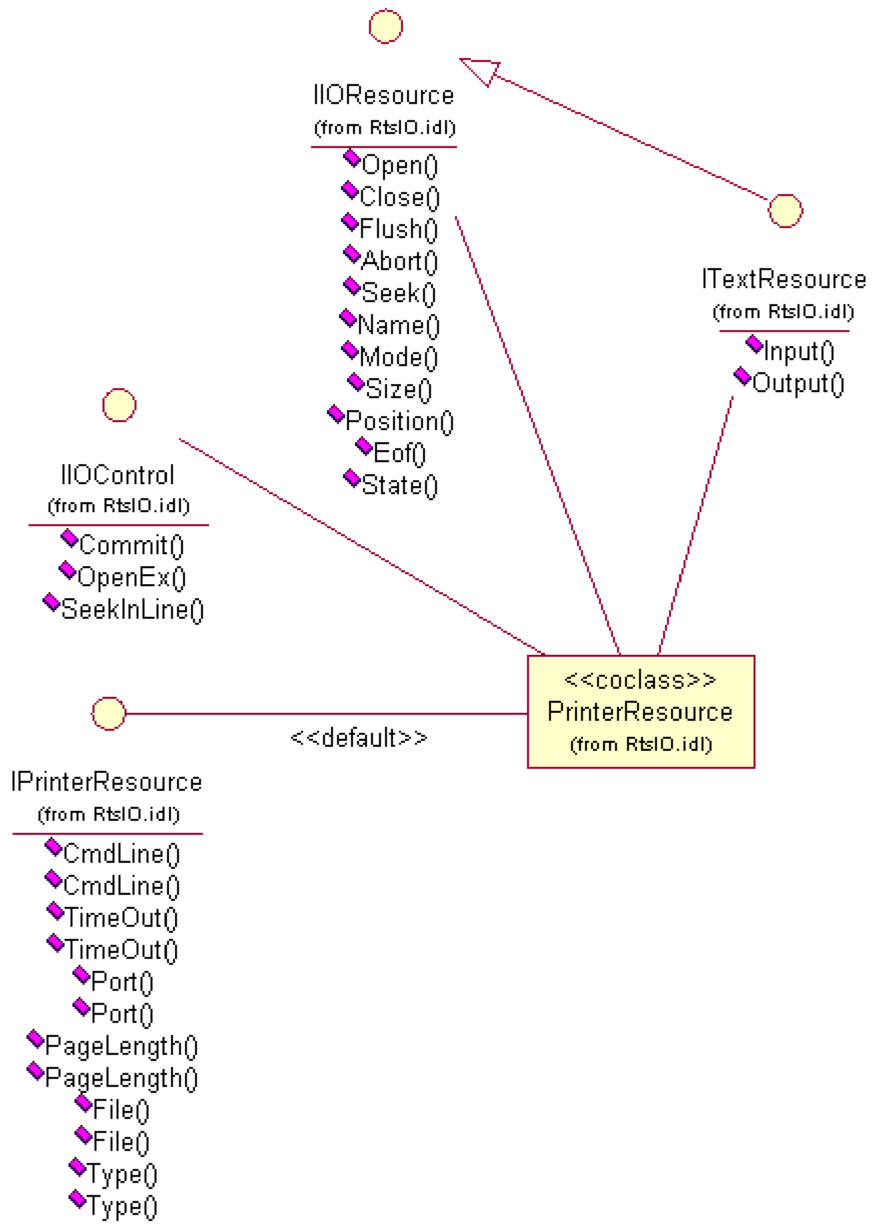
♦ ♦ Printing to a port, as if the printing is sent to a file, but the file name is a port name. The default port name is LPT1.

♦ ♦ Printing to a file in text/binary mode. The default file name is PRINTER.

Depending on which configuration is selected, the behavior of methods and properties of *IIOResource*, *ITextResource* or *IBinaryResource* may be different.

The specific interface for this resource is *IPrinterResource*. Its property *PrinterType* controls which method of printing is active: possible values are PRINTER_TYPE_WINDOWS, PRINTER_TYPE_PORT or PRINTER_TYPE_FILE.

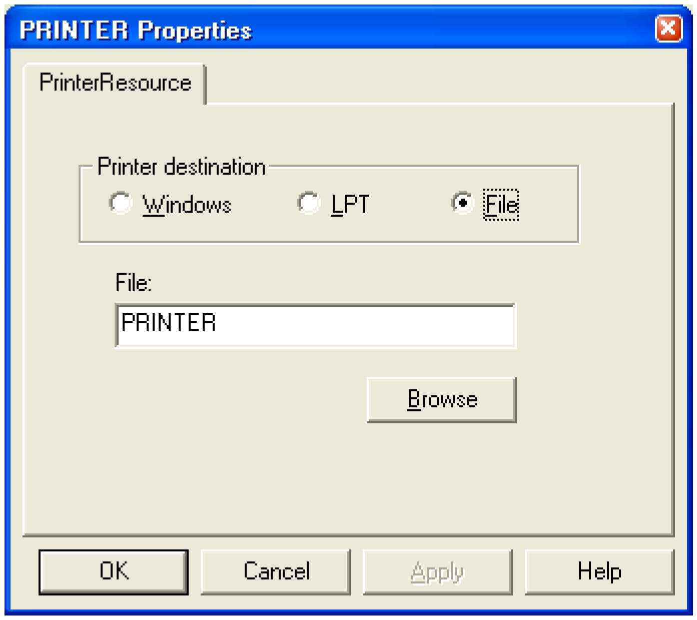
7.1 UML Description



7.2 IDL Description

```
[
	uuid(3F6B2D40-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("PrinterResource Class")
]
coclass PrinterResource
{
	[default] interface IPrinterResource;
	interface IIOResource;
	interface ITextResource;
	interface IIOControl;
};
```

7.3 Property Pages


PRINTER Properties

PrinterResource

Printer destination: Windows [ ] LPT [ ] File [●]

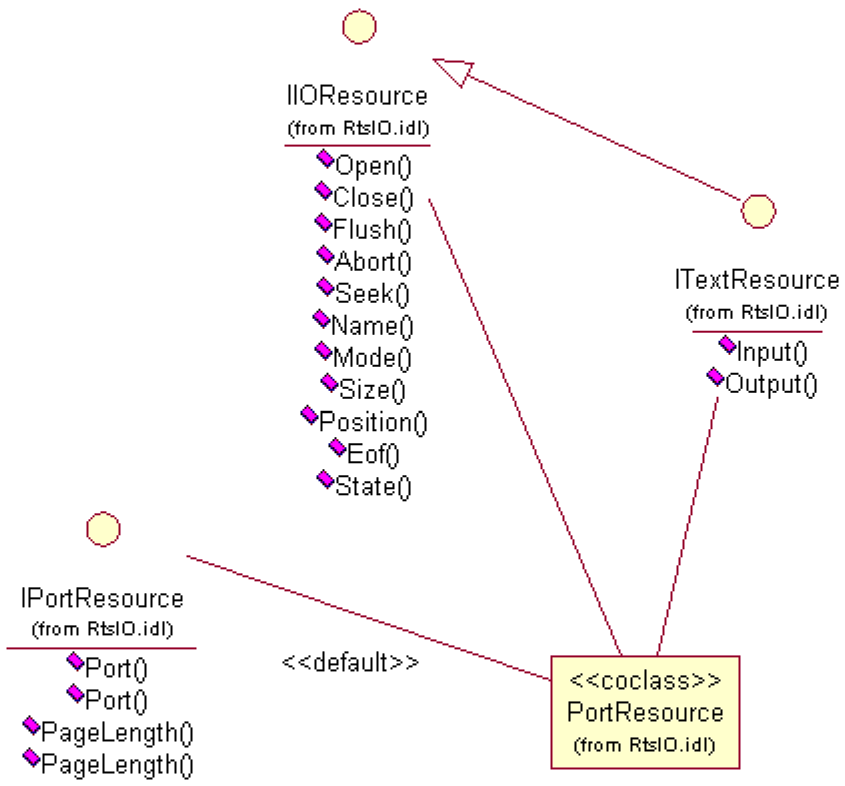
File: PRINTER

The display shows how to control the properties by clicking the “Properties” of the “PRINTER” Resource.

8 PortResource Component

The IPortResource is the default interface of the PortResource component.

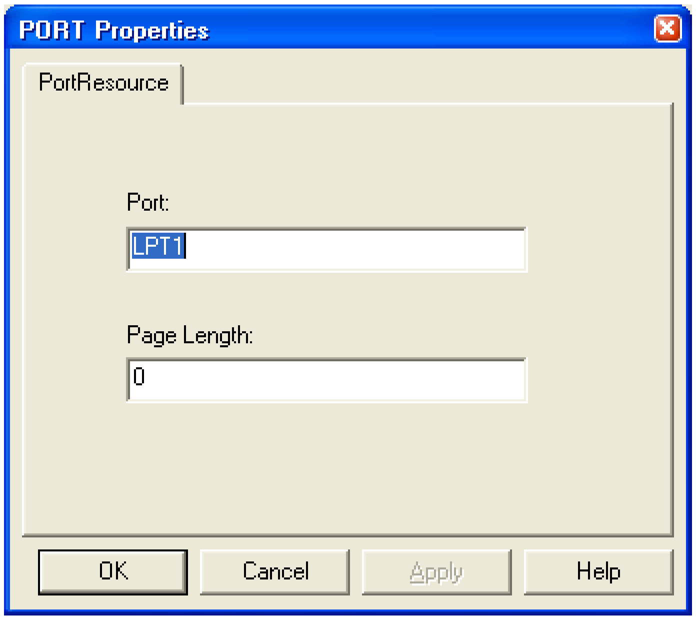
8.1 UML Description



8.2 IDL Description

```
[
	uuid(3F6B2D50-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("PortResource Class")
]
coclass PortResource
{
	[default] interface IPortResource;
	interface IIOResource;
	interface ITextResource;
};
```

8.3 Property Pages



Port: LPT1

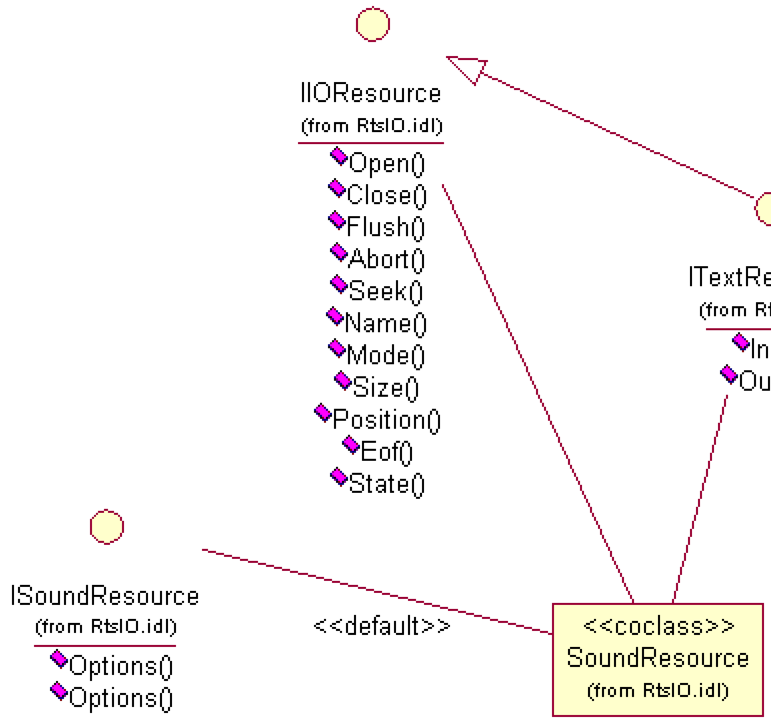
Page Length: 0

The display shows how to control the properties by clicking the “Properties” of the “PORT” Resource.

9 SoundResource Component

This resource is specialized in playing the sound files. It implements *ITextResource* and *IBinaryResource*. As a specific interface, *SoundResource* component implements *ISoundResource*.

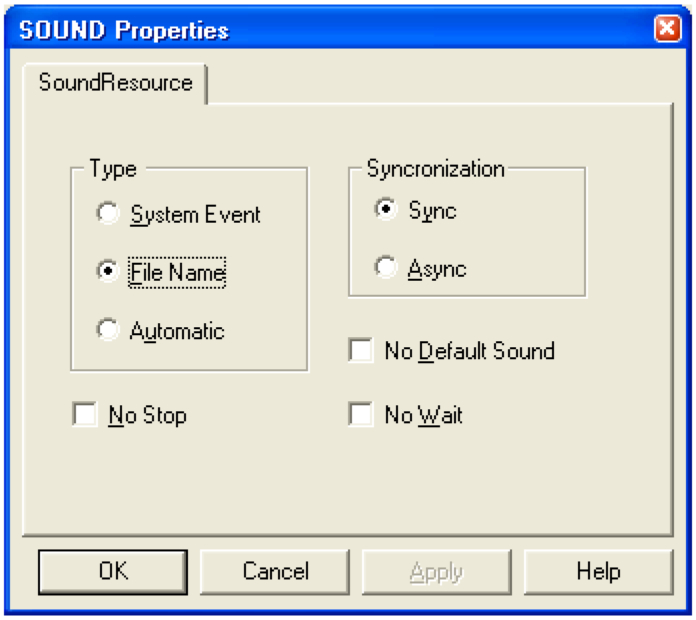
9.1 UML Description



9.2 IDL Description

```
[
        uuid(3F6B2D60-F0DA-11D2-BBB0-00C0268914D3),
        helpstring("SoundResource Class")
]
coclass SoundResource
{
        [default] interface ISoundResource;
        interface IIOResource;
        interface ITextResource;
};
```

9.3 Property Pages


SOUND Properties

SoundResource

Type: System Event [✓ File Name] Automatic

Syncronization: [✓ Sync] Async

☐ No Default Sound

☐ No Stop

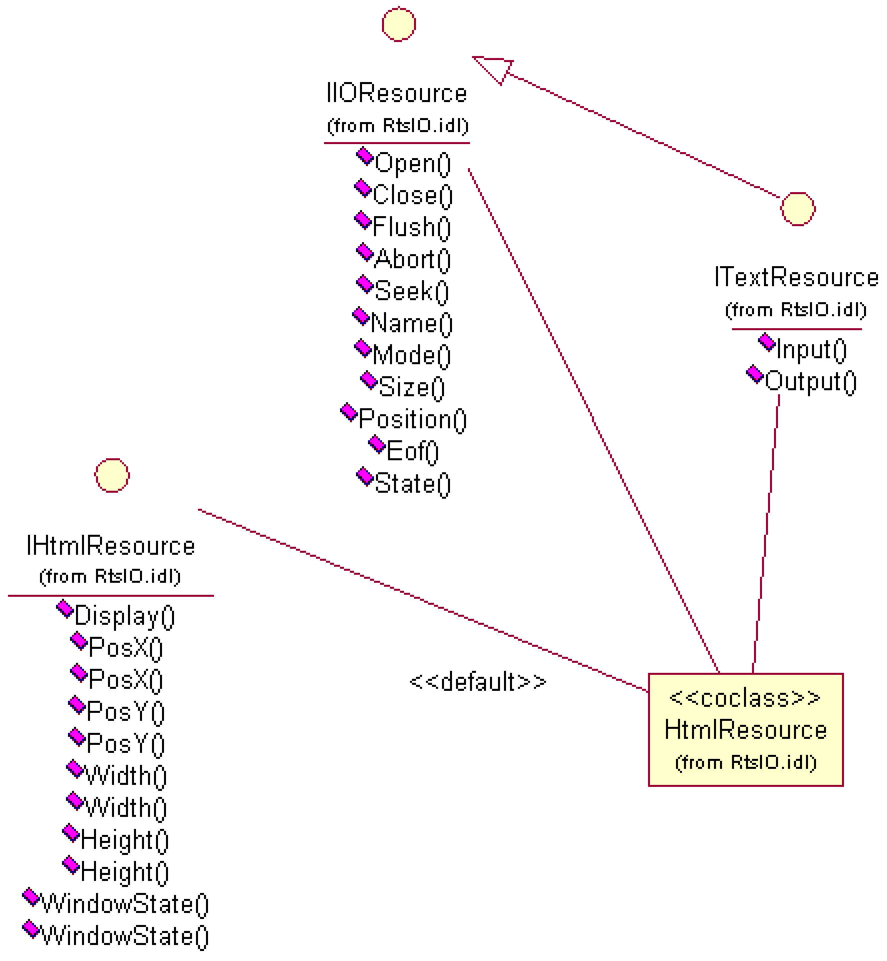
☐ No Wait

The display shows how to control the properties by clicking the “Properties” of the “SOUND” Resource.

10 HtmlResource Component

The *HtmlResource* component displays and renders a html resource (or URL). It implements the IHtmlResource as its default interface.

10.1 UML Description



10.2 IDL Description

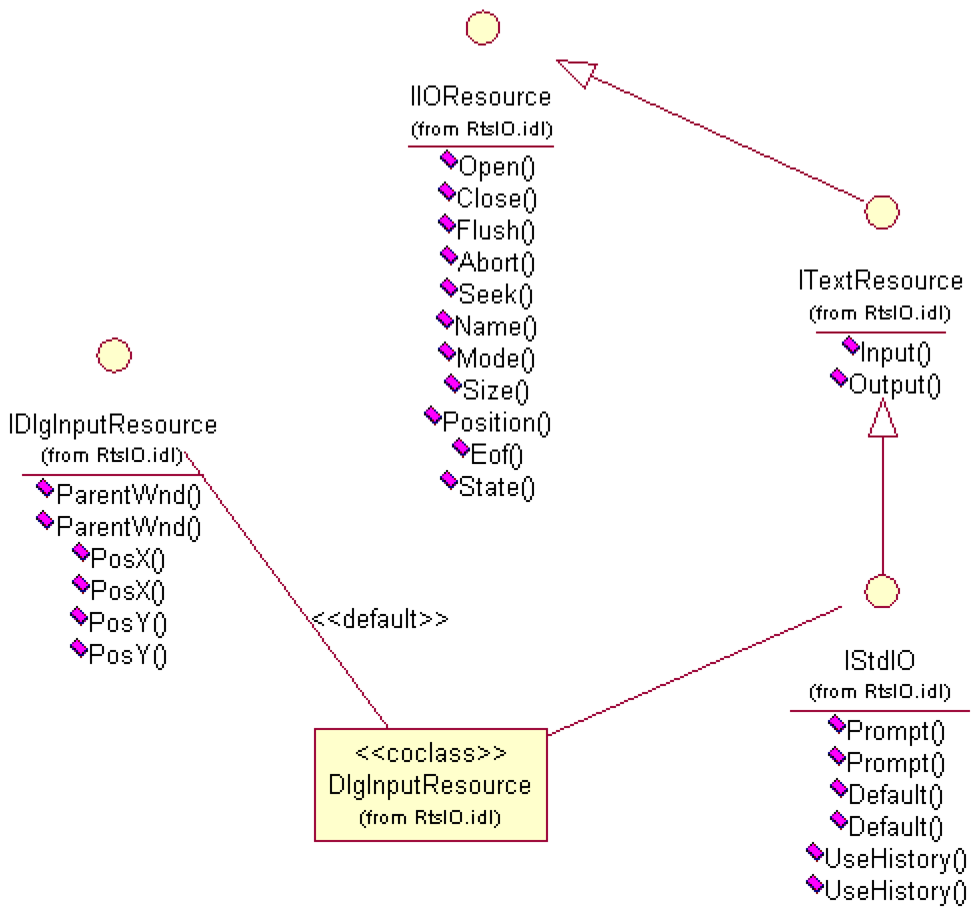
```
[
	uuid(3F6B2D70-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("HtmlResource Class")
]
coclass HtmlResource
{
	[default] interface IHtmlResource;
	interface IIOResource;
	interface ITextResource;
};
```

11 DlgInputResource Component

The *DlgInputResource* component displays different dialog boxes in which the user must enter the input value as a string or choose a Boolean value, depending on the input type.

The new specific interfaces for this resource are *IStdIO* and *IDlgInputResource. IStdIO* inherits *ITextResource.*

11.1 UML Description



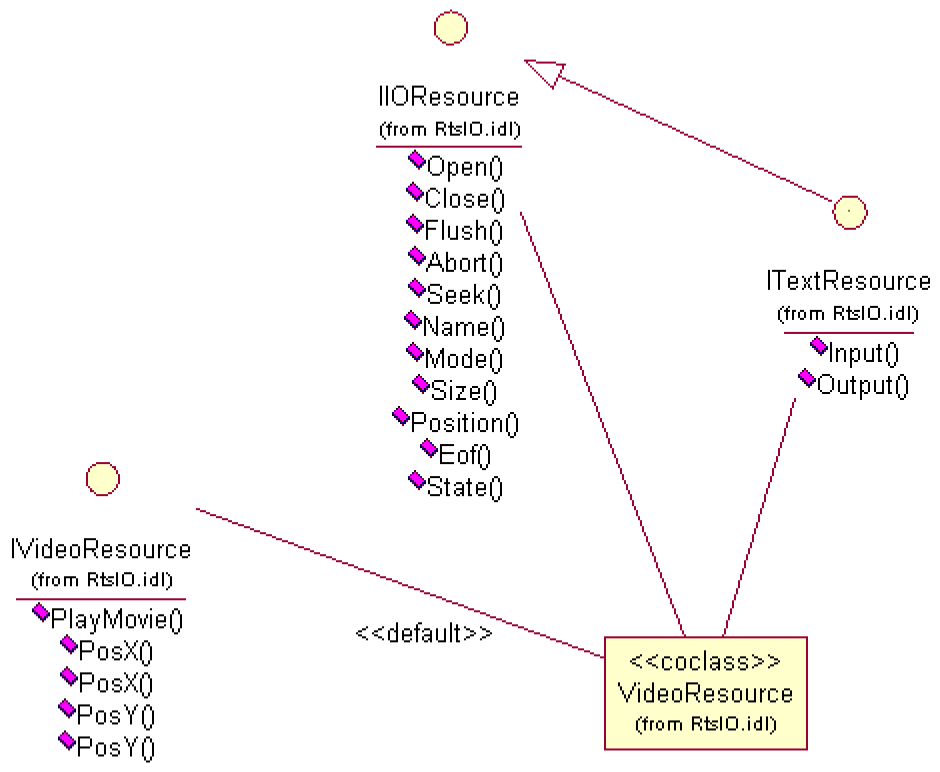
11.2 IDL Description

```
[
       uuid(3F6B2D80-F0DA-11D2-BBB0-00C0268914D3),
       helpstring("DlgInputResource Class")
]
coclass DlgInputResource
{
       [default] interface IDlgInputResource;
       interface IStdIO;
};
```

12 VideoResource Component

The *VideoResource* component displays and renders a video resource. It implements the IVideoResource as its default interface.

12.1 UML Description



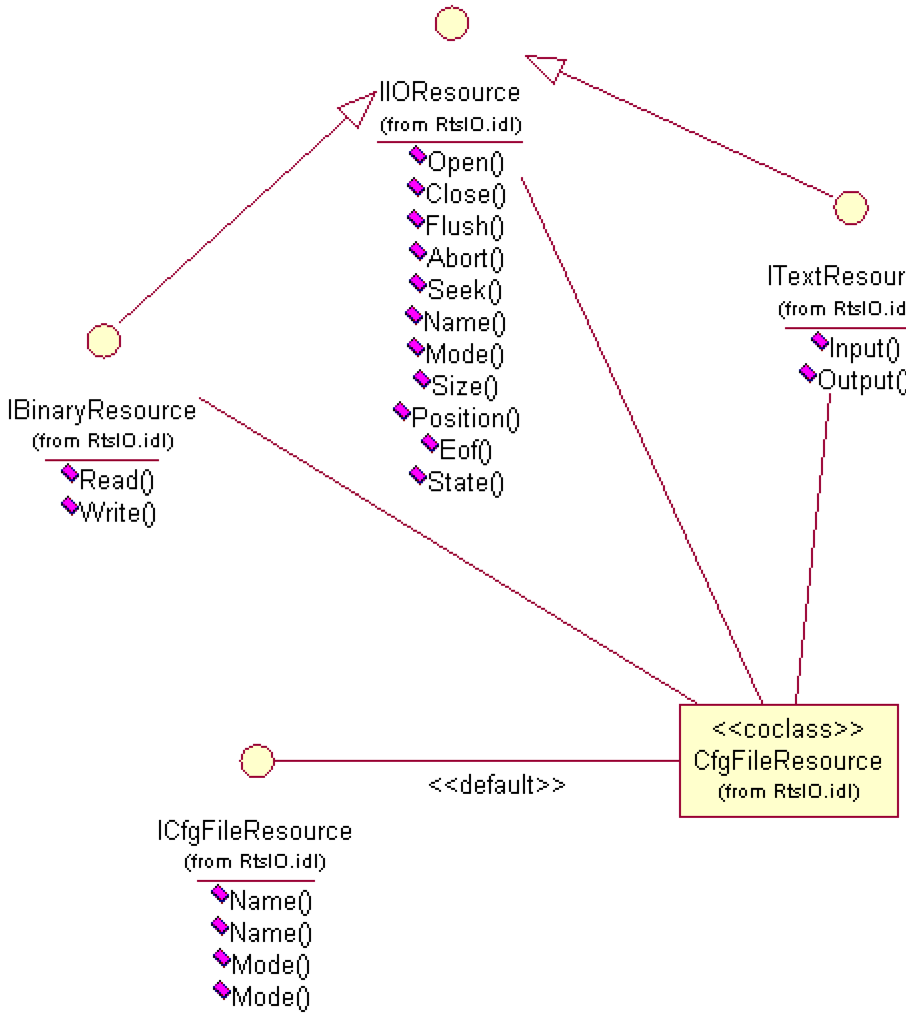
12.2 IDL Description

```
[
	uuid(3F6B2D90-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("VideoResource Class")
]
coclass VideoResource
{
	[default] interface IVideoResource;
	interface IIOResource;
	interface ITextResource;
};
```

13 CfgFileResource Component

It implements the ICfgFileResource as its default interface.

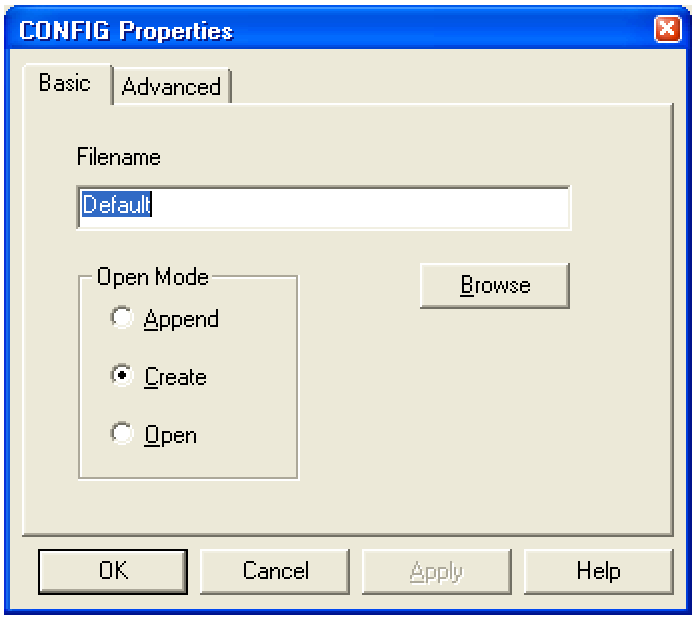
13.1 UML Description



13.2 IDL Description

```
[
	uuid(3F6B2DA0-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("CfgFileResource Class")
]
coclass CfgFileResource
{
	[default] interface ICfgFileResource;
	interface IBinaryResource;
	interface IIOResource;
	interface ITextResource;
};
```

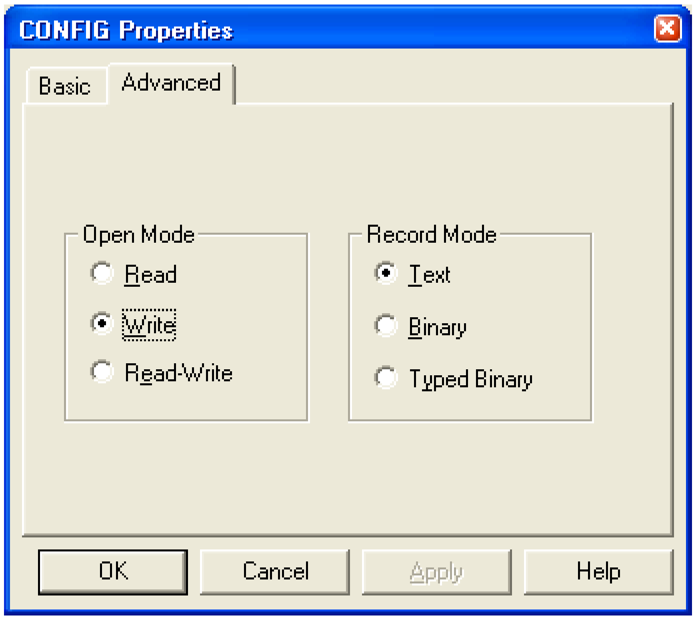
13.3 Property Pages


CONFIG Properties

Basic | Advanced

Filename: Default

Open Mode: Append [✓ Create] Open


CONFIG Properties

Basic | Advanced

Open Mode: Read [✓ Write] Read-Write

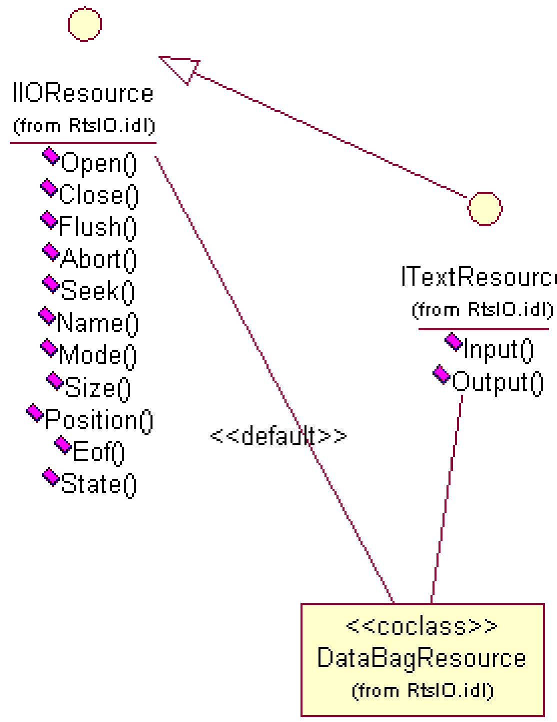
Record Mode: [✓ Text] Binary Typed Binary

The display shows how to control the “Basic” and “Advanced” properties by clicking the “Properties” of the “CONFIG” Resource.

14 DataBagResource Component

This resource stores strings, using an aggregated *DataContainer* object. The component implements *ITextResource* and exposes *IDataBag*, *IDataColl*, *and IDispatch* of the *DataContainer* object that it aggregates.

14.1 UML Description



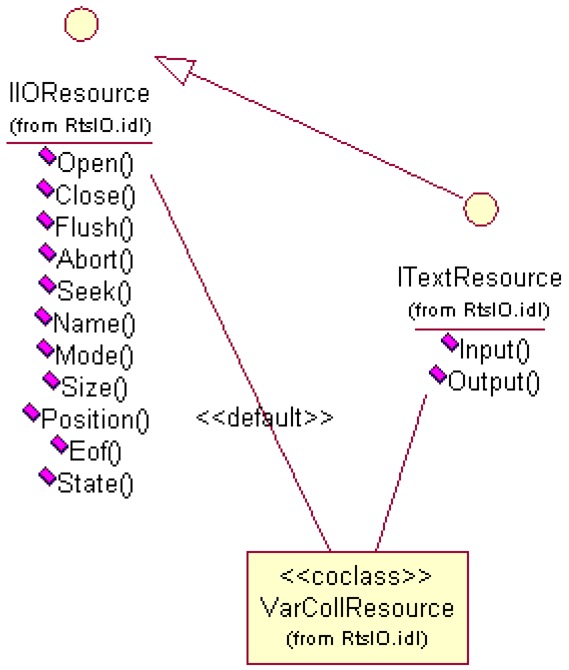
14.2 IDL Description

```
[
	uuid(3F6B2DB0-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("DataBagResource Class")
]
coclass DataBagResource
{
	[default] interface IIOResource;
	interface ITextResource;
};
```

15 VarCollResource Component

This resource stores strings, using an aggregated *VariantColl* object. The component implements *ITextResource* and exposes *IVariantColl and IDispatch* of the *VariantColl* object that it aggregates.

15.1 UML Description



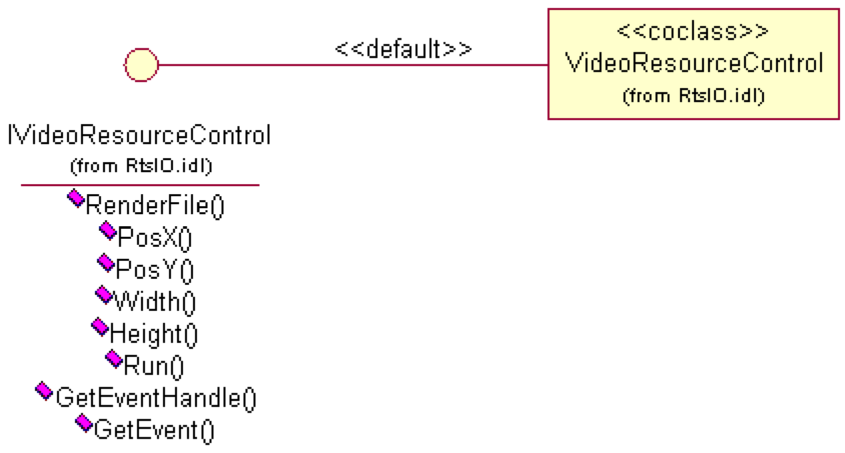
15.2 IDL Description

```
[
       uuid(3F6B2DC0-F0DA-11D2-BBB0-00C0268914D3),
       helpstring("VarCollResource Class")
]
coclass VarCollResource
{
       [default] interface IIOResource;
       interface ITextResource;
};
```

16 VideoResourceControl Component

It implements the IVideoResourceControl as its default interface.

16.1 UML Description



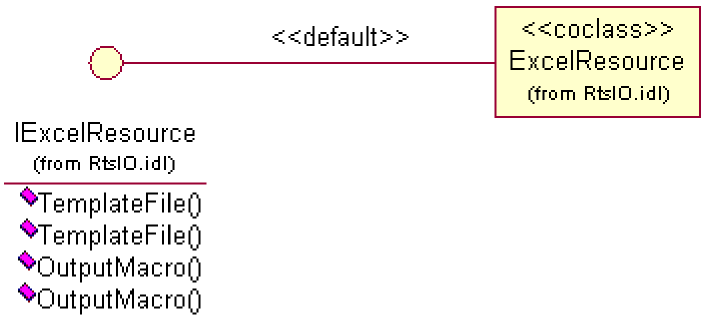
16.2 IDL Description

```
[
	uuid(3F6B2DD0-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("VideoResourceControl Class")
]
coclass VideoResourceControl
{
	[default] interface IVideoResourceControl;
};
```

17 ExcelResource Component

It implements the IExcelResource as its default interface.

17.1 UML Description

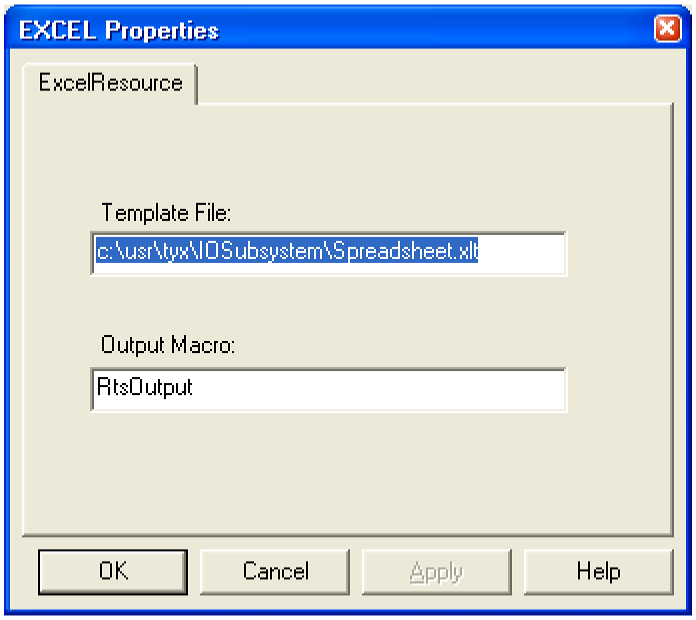


17.2 IDL Description

```
[
	uuid(3F6B2DE0-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("ExcelResource Class")
]
```

```
coclass ExcelResource
{
	[default] interface IExcelResource;
};
```

17.3 Property Pages



Template File: c:\usr\tyx\IOSubsystem\Spreadsheet.xlt

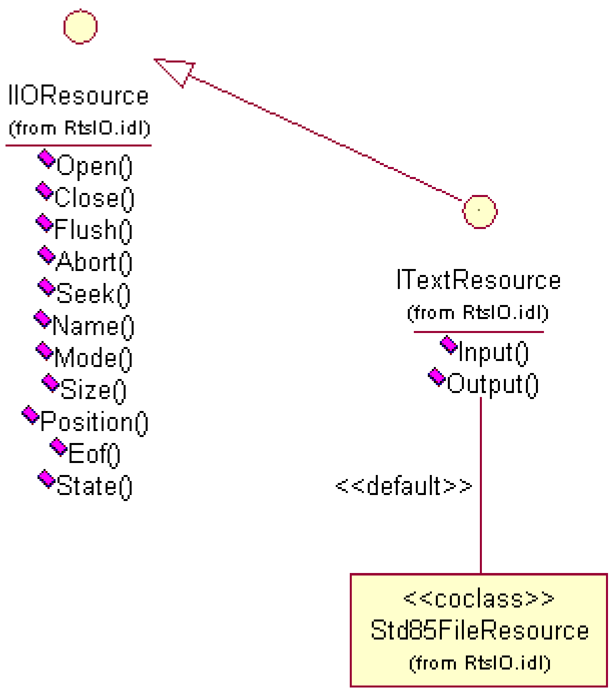
Output Macro: RtsOutput

The display shows how to control the properties by clicking the “Properties” of the “EXCEL” Resource.

18 Std85FileResource Component

It implements the ITextResource as its default interface.

18.1 UML Description



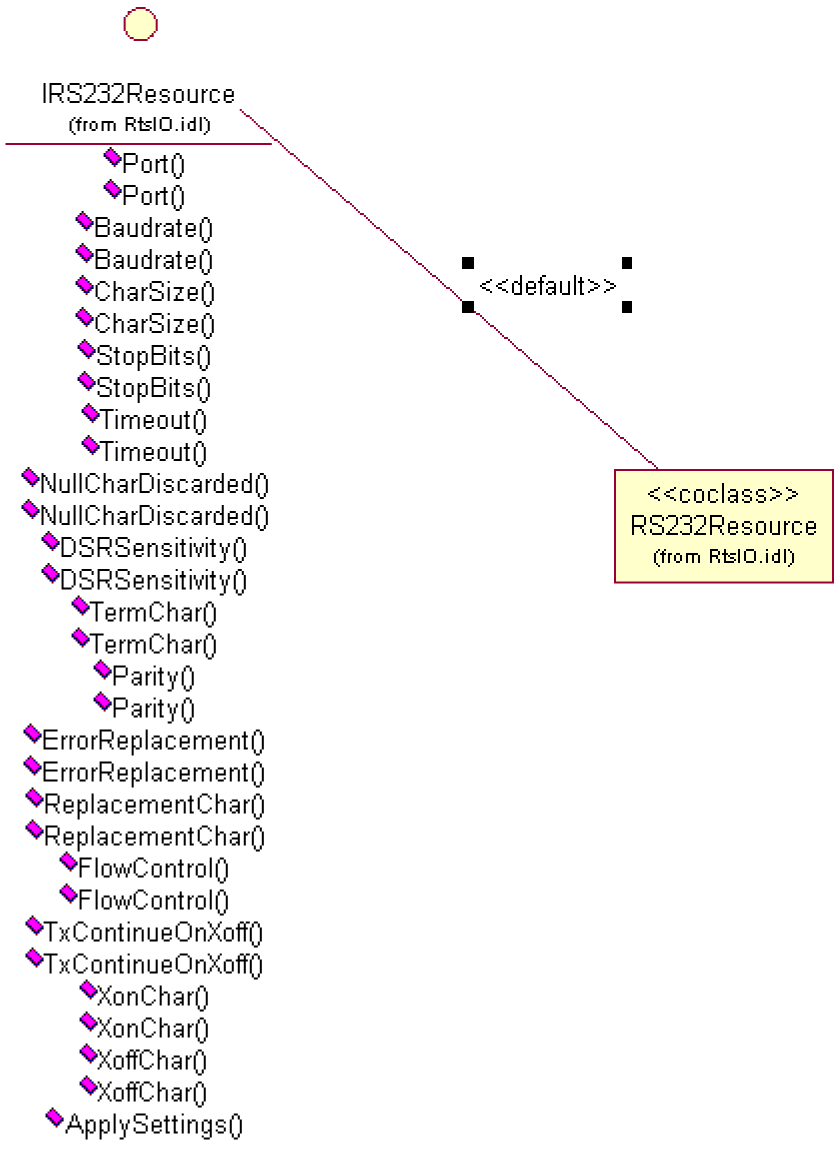
18.2 IDL Description

```
[
	uuid(3F6B2DF0-F0DA-11D2-BBB0-00C0268914D3),
	helpstring("Std85FileResource Class")
]
coclass Std85FileResource
{
	[default] interface ITextResource;
};
```

19 RS232Resource Component

It implements the IRS232Resource as its default interface.

19.1 UML Description



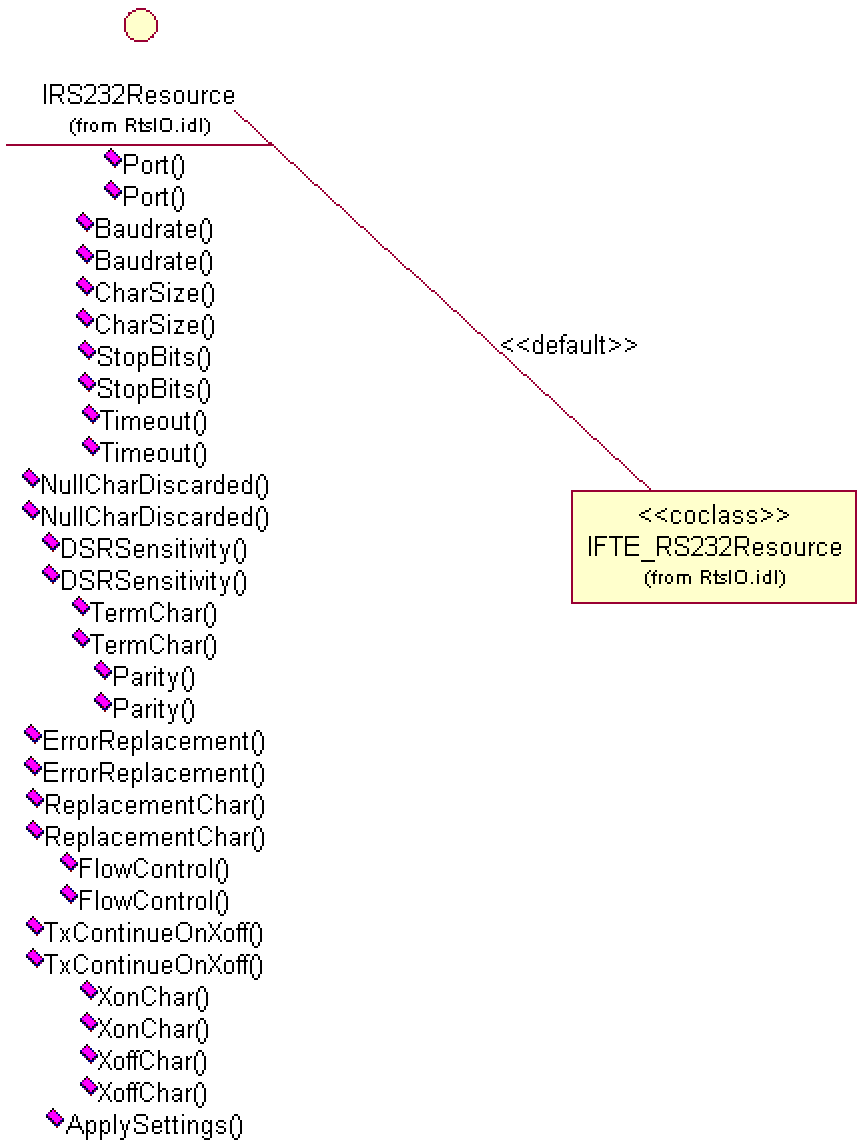
19.2 IDL Description

```
[
	uuid(3F6B2D10-00DA-11D2-BBB0-00C0268914D3),
	helpstring("RS232Resource Class")
]
coclass RS232Resource
{
	[default] interface IRS232Resource;
};
```

20 IFTE_RS232Resource Component

It implements the IRS232Resource as its default interface.

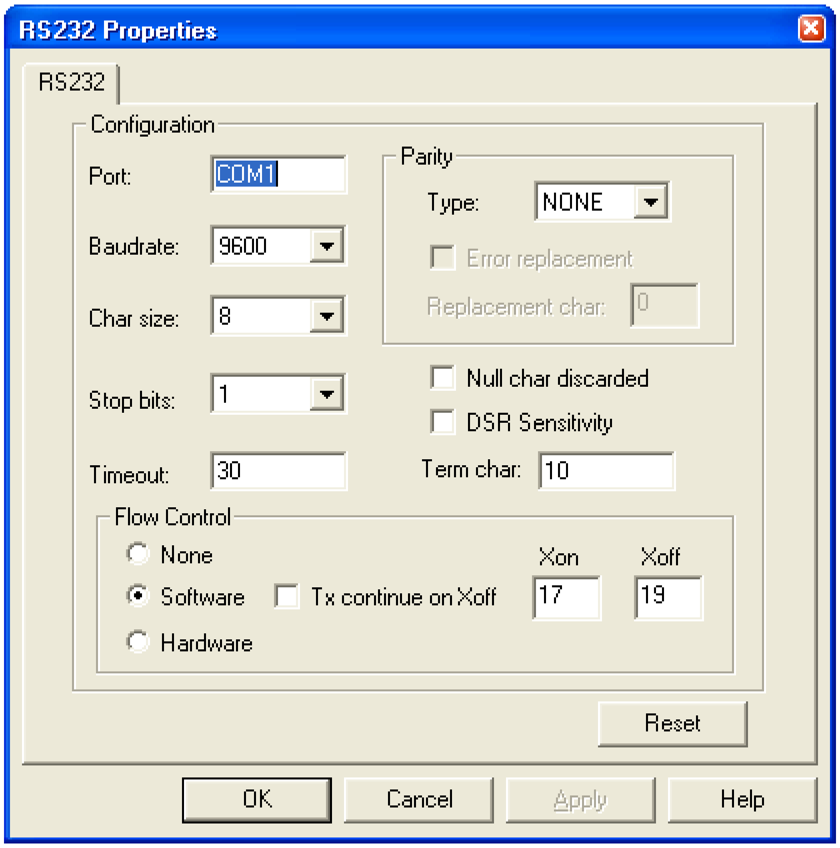
20.1 UML Description



20.2 IDL Description

```
[
	uuid(3F6B2D20-00DA-11D2-BBB0-00C0268914D3),
	helpstring("IFTE_RS232Resource Class")
]
coclass IFTE_RS232Resource
{
	[default] interface IRS232Resource;
};
```

20.3 Property Pages


RS232 Properties

RS232

Configuration

Port: COM1

Baudrate: 9600

Char size: 8

Stop bits: 1

Timeout: 30

Parity

Type: NONE

☐ Error replacement

Replacement char: 0

☐ Null char discarded

☐ DSR Sensitivity

Term char: 10

Flow Control

○ None

● Software ☐ Tx continue on Xoff

○ Hardware

Xon 17

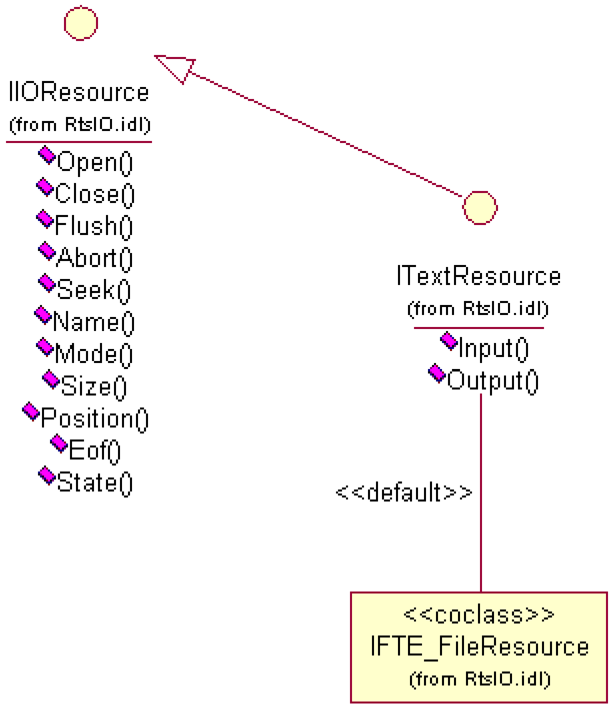
Xoff 19

The display shows how to control the properties by clicking the “Properties” of the “RS232” Resource.

21 IFTE_FileResource Component

It implements the ITextResource as its default interface.

21.1 UML Description



21.2 IDL Description

```
[
	uuid(3F6B2D30-00DA-11D2-BBB0-00C0268914D3),
	helpstring("IFTE_FileResource Class")
]
coclass IFTE_FileResource
{
	[default] interface ITextResource;
};
```

Note:
All additional interfaces, types and components referred by this document are described in the *COM Utils*, *TPS Server* or *IOSubsystem* documents.