FastReport Studio

Programmer's manual

Edition 1.08

Document revision history

1.00	August 26, 2005	Initial release
1.01	September 26, 2005	updates of export topic
1.02	September 26, 2005	Discuss the "Default connection" feature
1.03	September 29, 2005	Discuss multitasking issues
1.04	October 30, 2005	Minor updates
1.05	October 30, 2005	Minor updates
1.06	November 03, 2005	Minor updates
1.07	March 21, 2006	Objects hierarchy added
1.08	March 31, 2006	More detailed description on several topics

Table of Contents

FastReport objects review

FastReport Studio contains a number objects intended for report creation, modifying, exporting to different formats, and functionality enhancement. Let us explore some of the objects presented in FastReport Studio.

TfrxReport object

This object is the main one. One TfrxReport object contains one report. The object has all necessary properties and methods for report loading and saving, design and viewing.

Most significant methods of the TfrxReport object

HRESULT _stdcall ClearReport();

Clears a report.

HRESULT _stdcall LoadReportFromFile([in] BSTR szFileName);

Loads a report from a file with given name.

HRESULT _stdcall LoadReportFromStream([in] IUnknown* Stream);

Loads a report from the stream.

HRESULT _stdcall SaveReportToFile([in] BSTR FileName);

Saves a report to a file with given name.

HRESULT _stdcall SaveReportToStream([in] IUnknown* Stream);

Saves a report to the stream.

HRESULT _stdcall DesignReport();

Calls the report designer.

HRESULT _stdcall ShowReport();

Prepares a report and displays a result in the preview window.
Note: The ShowReport method does not compatible to ASP.NET solutions.

HRESULT _stdcall PrepareReport([in] VARIANT_BOOL ClearLastReport);

Starts a report without a preview window. If the "ClearLastReport" parameter is equal to "False," then a report will be added to the previously constructed one, otherwise the previously constructed report is cleared.

```
HRESULT _stdcall ShowPreparedReport();
```

Displays the report, which was a previously built via the "PrepareReport" call.

```
HRESULT _stdcall LoadPreparedReportFromFile([in] BSTR szFileName);
```

Loads the previously prepared and save report from file. This report can be freely displayed or exported.

```
HRESULT _stdcall SavePreparedReportToFile([in] BSTR szFileName);
```

Save prepared report into file in FP3 format.

```
HRESULT _stdcall PrintReport();
```

Prints a report. Print options can be set by the **PrintOptions** property of the report object.

Way to export

Following methods useful for export prepared report into number of external formats. These methods implemented as additional interface:

```
interface IfrxBuiltinExports : IUnknown
{

HRESULT _stdcall ExportToPDF(
      [in] BSTR FileName,
      [in] VARIANT_BOOL Compressed,
      [in] VARIANT_BOOL EmbeddedFonts,
      [in] VARIANT_BOOL PrintOptimized);
```

Exports prepared report to PDF format

```
HRESULT _stdcall ExportToXML(
      [in] BSTR FileName,
      [in] VARIANT_BOOL Styles,
      [in] VARIANT_BOOL PageBreaks,
      [in] VARIANT_BOOL WYSIWYG,
      [in] VARIANT_BOOL Background);
```

Exports prepared report to XML format

```
HRESULT _stdcall ExportToRTF(
      [in] BSTR FileName,
      [in] VARIANT_BOOL Pictures,
      [in] VARIANT_BOOL PageBreaks,
      [in] VARIANT_BOOL WYSIWYG);
```

Exports prepared report to the Rich Text Format

```
HRESULT _stdcall ExportToHTML(
      [in] BSTR FileName,
      [in] VARIANT_BOOL Pictures,
      [in] VARIANT_BOOL FixedWidth,
      [in] VARIANT_BOOL Multipage,
      [in] VARIANT_BOOL Navigator,
      [in] VARIANT_BOOL PicsInSameFolder,
      [in] VARIANT_BOOL Background);
```

```
Exports prepared report to HTML format
```

```
HRESULT _stdcall ExportToXLS(
      [in] BSTR FileName,
      [in] VARIANT_BOOL Pictures,
      [in] VARIANT_BOOL PageBreaks,
      [in] VARIANT_BOOL WYSIWYG,
      [in] VARIANT_BOOL AsText,
      [in] VARIANT_BOOL Background);
```

```
Exports prepared report to Microsoft Excel format
```

```
HRESULT _stdcall ExportToBMP(
      [in] BSTR FileName,
      [in] int Resolution,
      [in] VARIANT_BOOL Monochrome,
      [in] VARIANT_BOOL CropPages,
      [in] VARIANT_BOOL SeparatePages);
```

```
Exports prepared report to the Bitmap image[s]
```

```
HRESULT _stdcall ExportToJPEG(
      [in] BSTR FileName,
      [in] int Resolution,
      [in] int JpegQuality,
      [in] VARIANT_BOOL Monochrome,
      [in] VARIANT_BOOL CropPages,
      [in] VARIANT_BOOL SeparatePages);
```

```
Exports prepared report to the JPEG image[s]
```

```
HRESULT _stdcall ExportToTIFF(
      [in] BSTR FileName,
      [in] int Resolution,
      [in] VARIANT_BOOL Monochrome,
      [in] VARIANT_BOOL CropPages,
      [in] VARIANT_BOOL SeparatePages);
```

```
Exports prepared report to the TIFF images
```

```
HRESULT _stdcall ExportToTXT([in] BSTR FileName);
```

```
Exports prepared report to the simple ASCII text
```

```
HRESULT _stdcall ExportToCSV(
      [in] BSTR FileName,
      [in] BSTR Separator,
      [in] VARIANT_BOOL OEMCodepage);
```

```
Exports prepared report to the CSV format
```

```
};
```

Access to the Report Variables

The TfrxReport object provides methods for access to the report variables:

```
HRESULT _stdcall SetVariable(
      [in] BSTR Index,
      [in] VARIANT Value);
```

Sets variable’s value. Where **Index** is the variable name and **Value** is the value of variable.

```
HRESULT _stdcall GetVariable(
      [in] BSTR Index,
      [out, retval] VARIANT* Value);
```

Gets variable’s value

```
HRESULT _stdcall AddVariable(
      [in] BSTR Category,
      [in] BSTR Name,
      [in] VARIANT Value);
```

Add new variable

```
HRESULT _stdcall DeleteCategory([in] BSTR Name);
```

Delete category of variables

```
HRESULT _stdcall DeleteVariable([in] BSTR Name);
```
Deletes variable

Other useful methods of the TfrxReport object:

```
HRESULT _stdcall SelectDataset(
      [in] VARIANT_BOOL Selected,
      [in] IUnknown* DataSet);
```

This method has similar functionality of the designer menu: Report->Data->Select dataset. The **DataSet** argument can be one of the IfrxADOTable, IfrxADOQuery, or IfrxUserDataSet interfaces. If the **Selected** argument is true, then DataSet is enabled in report, otherwise disabled.

```
HRESULT _stdcall CreateReportObject(
      [in] IfrxComponent* ParentObject,
      [in] GUID ObjectType,
      [in] BSTR Name,
      [out, retval] IfrxComponent** GeneratedObject);
```

This method is useful for dynamic report creation. It provides functionality of creation of a report internal object and linking it to a parent object.

```
HRESULT _stdcall AddFunction(
                [in] BSTR FunctionDefinition,
                [in] BSTR Category,
                [in] BSTR Description);
```

This method registers a user defined function, which called by OnUserFunction event. The FunctionDefinition argument has the format of selected script language. Several possible examples follow:

Language	Examples of FunctuionDefinition argument
Pascal	`function SpellValue(Value: String): String`
	`function AnotherFn(Fisrt: Integer; Second: Etended): Extended`
C++	`int MyExternalfunction(int idx, char * str)`
	`double CalcSomething(double x, double y, double z)`

The Category argument allows logically group function in designer's list of available functions.

The Description argument set short description of function, which showing in designer tool.

Properties of the TfrxReport object

The TfrxReport object has the following properties:

```
HRESULT _stdcall EngineOptions([out, retval] IfrxEngineOptions** Value);
```

A set of properties related to the FastReport engine.

```
HRESULT _stdcall PreviewOptions([out, retval] IfrxPreviewOptions** Value);
```

A set of properties related to the report preview.

```
HRESULT _stdcall PrintOptions([out, retval] IfrxPrintOptions** Value);
```

A set of properties related to the report printing.

```
HRESULT _stdcall ReportOptions([out, retval] IfrxReportOptions** Value);
```

A set of properties related to the report.

```
HRESULT _stdcall Resources([out, retval] IfrxResources** Value);
```

A set of properties related to the engine resources.

```
HRESULT _stdcall ScriptLanguage([out, retval] BSTR* Value);
HRESULT _stdcall ScriptLanguage([in] BSTR Value);
```

Script language used in a report.

```
HRESULT _stdcall ScriptText([out, retval] BSTR* Value);
HRESULT _stdcall ScriptText([in] BSTR Value);
```

Give access to the report script. This script executes at report preparation time.

```
HRESULT _stdcall Errors([out, retval] BSTR* Value);
```

The CR\LF terminated string of errors, occurring during one or another operation.

```
HRESULT _stdcall MainWindowHandle([in] long hWnd);
```

Provides write-only access to the window handle of the main window of the application. Use this property to set a parent window for designer and viewer. This property makes such windows part of the application, so that they are minimized, restored, enabled, and disabled with the application.

A set of properties related to the FastReport engine:

```
interface IfrxEngineOptions : IUnknown
{

HRESULT _stdcall ConvertNulls([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall ConvertNulls([in] VARIANT_BOOL Value);
```

Converts the "Null" value of the DB field into "0," "False," or empty string, depending on the field type.

```
HRESULT _stdcall DoublePass([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall DoublePass([in] VARIANT_BOOL Value);
```

Makes a report a two-pass one.

```
HRESULT _stdcall MaxMemSize([out, retval] int* Value);
HRESULT _stdcall MaxMemSize([in] int Value);
```

The maximum size of memory in Mbytes, allocated to the report pages' cache. It becomes useful in cases when the "UseFileCashe" property is equal to "True." If a report begins to occupy more memory during construction, caching of the constructed report pages into a temporary file is performed. This property is inexact and allows only approximate determination of the memory limit.

```
HRESULT _stdcall PrintIfEmpty([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall PrintIfEmpty([in] VARIANT_BOOL Value);
```

Defines, whether it is necessary to print a blank report (one which containing no data lines).

```
HRESULT _stdcall SilentMode([out, retval] frxSilentMode* Value);
HRESULT _stdcall SilentMode([in] frxSilentMode Value);
```

"Silent" mode. Thus all messages about errors are stored in the "TfrxReport.Errors" property, without displaying any messages on the screen.

```
HRESULT _stdcall TempDir([out, retval] BSTR* Value);
HRESULT _stdcall TempDir([in] BSTR Value);
```

Specifies a path to the directory for storing temporary files.

```
HRESULT _stdcall UseFilecache([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall UseFilecache([in] VARIANT_BOOL Value);
```

Defines, whether it is necessary to use report pages caching into the file (see the "MaxMemSize" property).

```
};
```

A set of properties, relating to the report preview:

```
interface IfrxPreviewOptions : IUnknown
{

HRESULT _stdcall AllowEdit([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall AllowEdit([in] VARIANT_BOOL Value);
```

Enables or disables a finished report editing.

```
HRESULT _stdcall Buttons([out, retval] frxPreviewButton* Value);
HRESULT _stdcall Buttons([in] frxPreviewButton Value);
```

A set of buttons, which will be available in the preview window.

The available values of this property are the following:

pbPrint - printing
pbLoad – loading from a file
pbSave – saving into a file
pbExport - export
pbZoom - zooming
pbFind - search
pbOutline – report outline enabling
pbPageSetup – page properties
pbTools - tools
pbEdit - editor
pbNavigator - navigation

You can combine any of these values.

```
HRESULT _stdcall DoubleBuffered([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall DoubleBuffered([in] VARIANT_BOOL Value);
```

A double-buffer mode for the preview window. If enabled (by default), the preview window will not flicker during repainting, but the process speed would be reduced.

```
HRESULT _stdcall Maximazed([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall Maximazed([in] VARIANT_BOOL Value);
```

Defines whether the preview window is maximized.

```
HRESULT _stdcall MDIChild([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall MDIChild([in] VARIANT_BOOL Value);
```

Defines whether the preview window is MDIChild (for MDI interface organizing).

```
HRESULT _stdcall Modal([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall Modal([in] VARIANT_BOOL Value);
```

Defines whether the preview window is modal.

```
HRESULT _stdcall OutlineExpand([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall OutlineExpand([in] VARIANT_BOOL Value);
```

Defines whether the report outline is expanded or not.

```
HRESULT _stdcall OutlineVisible([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall OutlineVisible([in] VARIANT_BOOL Value);
```

Defines whether the panel with the report outline is visible.

```
HRESULT _stdcall OutlineWidth([out, retval] int* Value);
HRESULT _stdcall OutlineWidth([in] int Value);
```

Defines width of the panel with the report outline.

```
HRESULT _stdcall ShowCaptions([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall ShowCaptions([in] VARIANT_BOOL Value);
```

Defines whether it is necessary to display button captions. When enabling this property, you should limit the number of the displayed buttons in the Buttons property, since all the buttons would not find room on the screen.

```
HRESULT _stdcall Zoom([out, retval] double* Value);
HRESULT _stdcall Zoom([in] double Value);
```

The default zooming value.

```
HRESULT _stdcall ZoomMode([out, retval] frxZoomMode* Value);
HRESULT _stdcall ZoomMode([in] frxZoomMode Value);
```

Default zooming mode. The following values are available:

zmDefault – zooming via the “Zoom” property;
zmWholePage – the whole page fits;
zmPageWidth – the page width fits;
zmManyPages – two pages fit.

```
};
```

A set of properties, which relate to the report printing:

```
interface IfrxPrintOptions : IUnknown
{
HRESULT _stdcall Copies([out, retval] int* Value);
HRESULT _stdcall Copies([in] int Value);
```

A number of the printable copies by default.

```
HRESULT _stdcall Collate([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall Collate([in] VARIANT_BOOL Value);
```

Whether to collate the copies.

```
HRESULT _stdcall PageNumbers([out, retval] BSTR* Value);
HRESULT _stdcall PageNumbers([in] BSTR Value);
```

Page numbers, which are to be printed. For example, “1,3,5-12,17-“.

```
HRESULT _stdcall Printer([out, retval] BSTR* Value);
HRESULT _stdcall Printer([in] BSTR Value);
```

Printer name.

```
HRESULT _stdcall PrintPages([out, retval] frxPrintPages* Value);
HRESULT _stdcall PrintPages([in] frxPrintPages Value);
```

Defines the pages to be printed. The following values are available:
ppAll - all
ppOdd - odd
ppEven - even

```
HRESULT _stdcall ShowDialog([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall ShowDialog([in] VARIANT_BOOL Value);
```

Whether to display a print dialogue.

```
};
```

A set of properties relating to the report:

```
interface IfrxReportOptions : IUnknown
{
```

```
HRESULT _stdcall Author([out, retval] BSTR* Value);
HRESULT _stdcall Author([in] BSTR Value);
```

Report author.

```
HRESULT _stdcall Compressed([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall Compressed([in] VARIANT_BOOL Value);
```

Whether to store reports and prepared reports in compressed GZIP format.

```
HRESULT _stdcall ConnectionName([out, retval] BSTR* Value);
HRESULT _stdcall ConnectionName([in] BSTR Value);
```

The name of default connection. Default connection is used for TfrxADOTable and TfrxADOQuery if no TfrxADOConnection is present.

```
HRESULT _stdcall CreationDate([out, retval] DATE* Value);
HRESULT _stdcall CreationDate([in] DATE Value);
```

Report creation date.

```
HRESULT _stdcall Description([out, retval] BSTR* Value);
HRESULT _stdcall Description([in] BSTR Value);
```

Report description.

```
HRESULT _stdcall Name([out, retval] BSTR* Value);
HRESULT _stdcall Name([in] BSTR Value);
```

Report name.

```
HRESULT _stdcall LastChange([out, retval] DATE* Value);
HRESULT _stdcall LastChange([in] DATE Value);
```

The date the report was last modified.

```
HRESULT _stdcall Password([out, retval] BSTR* Value);
HRESULT _stdcall Password([in] BSTR Value);
```

Report password. If this property is not blank, a password is required when opening a report.

```
HRESULT _stdcall VersionBuild([out, retval] BSTR* Value);
HRESULT _stdcall VersionBuild([in] BSTR Value);
HRESULT _stdcall VersionMajor([out, retval] BSTR* Value);
HRESULT _stdcall VersionMajor([in] BSTR Value);
HRESULT _stdcall VersionMinor([out, retval] BSTR* Value);
HRESULT _stdcall VersionMinor([in] BSTR Value);
HRESULT _stdcall VersionRelease([out, retval] BSTR* Value);
HRESULT _stdcall VersionRelease([in] BSTR Value);
```

```
 These properties define the number of a version.
};
```

A set of properties relating to the engine resources:

```
interface IfrxResources : IUnknown
{

HRESULT _stdcall HelpFile([out, retval] BSTR* Value);
HRESULT _stdcall HelpFile([in] BSTR Value);
```

Set/get path to the help file.

```
HRESULT _stdcall Help();
```

Show help.

```
HRESULT _stdcall GetResourceString(
      [in] BSTR ID,
      [out, retval] BSTR* ResourceStr);
```

This method is useful to get a resource string for a resource, which identified by the **ID** argument. Resource language depends on selected language.

```
HRESULT _stdcall LoadLanguageResourcesFromFile([in] BSTR FileName);
```

This method is useful for selection of the national language for user interface of Designer and Viewer. The **FileName** argument must be a full path to the file in the FastReport language resource format (files with .frc extension). The default language set to English.

```
};
```

The following events are defined in the TfrxReport object:

```
interface IfrxReportEvents : IDispatch
{

HRESULT OnAfterPrint([in] IfrxComponent* Sender);
```

Event occurs after handling each object.

```
HRESULT OnBeforePrint([in] IfrxComponent* Sender);
```

Event occurs before handling each object.

```
HRESULT OnClickObject( [in] IfrxView* Sender, [in] long Button);
```

This event occurs when user clicks on the object in the preview window. The Sender argument is the interface on clicked object. See description below. The Button argument is the identifier of clicked the mouse button.

```
HRESULT OnUserFunction(
      [in] BSTR MethodName,
      [in] VARIANT Params,
      [out, retval] VARIANT* ResultValue);
```

Event occurs when calling a function, added with the help of the “AddFunction” method. See more in the corresponding chapter.

```
HRESULT OnBeginDoc([in] IfrxComponent* Sender);
```

Event occurs before build report.

```
HRESULT OnEndDoc([in] IfrxComponent* Sender);
```

Event occurs on complete build report.

```
HRESULT OnPrintReport([in] IfrxComponent* Sender);
```

Event occurs on complete printing report.

```
HRESULT OnProgress(
      [in] IfrxReport* Sender,
      [in] long ProgressType,
      [in] int Progress);
```

This event periodically occurs to inform an application about progress of report processing.

```
HRESULT OnProgressStart();
```

Signals start report processing

```
HRESULT OnProgressStop();
```

Signals the completion or abort report porcessing
};

TfrxUserDataset

Object for data access. The FastReport Studio uses this object for navigation and reference to the data set fields. The TfrxUserDataSet object allows constructing reports, which are not related to the database data, but do receive data from other sources (for example, array, file, etc.). At the same time, a programmer should provide navigation in such source. (See events below).

Properties

The TfrxUserDataSet component has the following properties:

```
interface IfrxUserDataSet : IUnknown
{

HRESULT _stdcall Fields([out, retval] BSTR* Value);
HRESULT _stdcall Fields([in] BSTR Value);
```

CR/LF terminated list of UserDataSet fields.

```
HRESULT _stdcall Name([out, retval] BSTR* Value);
HRESULT _stdcall Name([in] BSTR Value);
```

A symbolic name, under which the dataset will be displayed in the designer.

```
};

Additional interface intended for TfrxUserDataSet object.

interface IfrxDataSet : IUnknown
{

HRESULT _stdcall UserName([out, retval] BSTR* Value);
HRESULT _stdcall UserName([in] BSTR Value);

The name of the dataset

HRESULT _stdcall RangeBegin([out, retval] frxRangeBegin* Value);
HRESULT _stdcall RangeBegin([in] frxRangeBegin Value);

The navigation start point. The following values are available:
rbFirst – from the beginning of the data
rbCurrent – from the current record

HRESULT _stdcall RangeEndCount([out, retval] int* Value);
HRESULT _stdcall RangeEndCount([in] int Value);

A count of records in the data set, if the "RangeEnd" property = reCount.

HRESULT _stdcall RangeEnd([out, retval] frxRangeEnd* Value);
HRESULT _stdcall RangeEnd([in] frxRangeEnd Value);

The endpoint of navigation. The following values are available:
reLast – till the end of the data
reCurrent – till the current record
reCount – by the number of records set in the "RangeEndCount" property

};
```

Events

```
The following events are defined for the TfrxUserDataSet object:

interface IfrxUserDataSetEvents : IDispatch
{
HRESULT OnGetValue(
      [in] VARIANT VarName,
      [out] VARIANT* Value);

This event's handler must return the Value of the VarName variable.

HRESULT OnCheckEOF([out] VARIANT_BOOL* IsEOF);
```

This event's handler must return the Eof = True parameter, if the end of the data set is reached.

HRESULT OnFirst();

This event's handler must move the cursor to the beginning of the data set.

HRESULT OnNext();

This event's handler must move the cursor to the next record.

HRESULT OnPrior();

This event's handler must move the cursor to the previous record.

};

ADO Components

ADO components are set of add-in object, which provides access to the ADO engine. It contains the following objects: “TfrxADODatabase”, “TfrxADOTable” and “TfrxADOQuery”.

TfrxADODatabase object

TfrxADOConnection encapsulates the ADO connection object. Use TfrxADOConnection for connecting to ADO data stores. The connection provided by a single TfrxADOConnection object can be shared by multiple TfrxADOTable and TfrxADOQuery objects through their DataBase properties.

```
HRESULT _stdcall ConnectionString([out, retval] BSTR* Value);
HRESULT _stdcall ConnectionString([in] BSTR Value);
```

Set ConnectionString to specify the information needed to connect the ADO connection object to the data store. The value used for ConnectionString consists of one more arguments ADO uses to establish the connection.

```
HRESULT _stdcall LoginPrompt([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall LoginPrompt([in] VARIANT_BOOL Value);
```

Specifies whether a login dialog appears immediately before opening a new connection.

```
HRESULT _stdcall Connected([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall Connected([in] VARIANT_BOOL Value);
```

Specifies whether the connection is active. Set Connected to true to establish a connection to an ADO data store. Set Connected to false to close a connection. The default value for Connected is false.

An application can check Connected to determine the current status of a connection. If Connected is true, the connection is active; if false, then the connection is inactive.

TfrxADOTable object

TfrxADOTable is a dataset component that encapsulates a table accessed through an ADO data store. This table used as data source for report building.

```
HRESULT _stdcall DataBase([out, retval] IfrxADODatabase** Value);
HRESULT _stdcall DataBase([in] IfrxADODatabase* Value);
```

Set/read the TfrxADODatabase object. Useful for link the TfrxADODatabase object to the IfrxADOQuery object.

```
HRESULT _stdcall IndexName([out, retval] BSTR* Value);
HRESULT _stdcall IndexName([in] BSTR Value);
```

Specifies the currently active index. Use IndexName to activate an index and cause it to actively order the dataset's rows. At runtime, set IndexName to a string containing the name of the index.

```
HRESULT _stdcall TableName([out, retval] BSTR* Value);
HRESULT _stdcall TableName([in] BSTR Value);
```

Indicates the base table operated on. Use TableName to specify the base table in a database on which the ADO table component operates.

```
HRESULT _stdcall Name([out, retval] BSTR* Value);
HRESULT _stdcall Name([in] BSTR Value);
```

A symbolic name, under which the dataset will be displayed in the designer.

TfrxADOQuery object

TADOQuery provides the means for issuing SQL against an ADO data store and showing results on report pages.

```
HRESULT _stdcall DataBase([out, retval] IfrxADODatabase** Value);
HRESULT _stdcall DataBase([in] IfrxADODatabase* Value);
```

Set/read the TfrxADODatabase object. Useful for link the TfrxADODatabase object to the IfrxADOQuery object.

```
HRESULT _stdcall Query([out, retval] BSTR* Value);
HRESULT _stdcall Query([in] BSTR Value);
```

Contains the text of the SQL statement to execute for the ADO query.

```
HRESULT _stdcall Name([out, retval] BSTR* Value);
HRESULT _stdcall Name([in] BSTR Value);
```

A symbolic name, under which the dataset will be displayed in the designer.

Working with TfrxReport object

Configuring and setting environment

The way of creation report object depends on platform and programming language. So, we will describe it for Visaual C++, C# and Visual Basic.

Visual C++

We suggest using ATL for report development. Following code shows example of report object creation for Visual C++:

```
#if _MSC_VER < 1300
  // Use this for MS Visual Studio 6.
  #import "path\\to\\FastReport3.dll" named_guids auto_rename
#else
  // This code preffered for MS Visual Studio.NET
  #import "libid:d3c6fb9b-9edf-48f3-9a02-6d8320eaa9f5" named_guids auto_rename
#endif

using namespace FastReport;
```

C#.NET and Visual Basic.NET

For using FastReport with C#.Net it is need to set reference to the FastReport library. It can be done by Add Reference menu, which appears by left clicking on the Reference item in the Solution explorer window. Select COM tab in the Add Reference window. Locate the FastReport3 report generator component and select it. Finally, press Ok button.

Creation of the TfrxReport object

Visual C++

Use header described in previous chapter. After that you'd able to use following code inside function body. Internally, it uses ATL AutoPtr functionality:

```
      IfrxReportPtr    pReport(__uuidof(TfrxReport));
```

C#.NET

Use steps described in previous chapter. After that you'd able to use following code to create report object:

```
   TfrxReportClass     report;
   report = new TfrxReportClass();
```

Visual Basic.NET

Use steps described in previous chapter. After that you'd able to use following code to create report object:

```
    Dim frx As New TfrxReportClass()
```

Loading and saving a report

Further examples are almost same for all platforms, including Visual C++, Visual C#.NET, Visual Basic.NET, and others. Use pointer addressing for C++.

Visual C++

```
// Loads report from file "1.fr3"
pReport->LoadReportFromFile("c:\1.fr3");
// Saves report to file "1.fr3"
pReport->SaveReportToFile("c:\2.fr3");
```

C#.NET, Visual Basic.NET, and others

```
// Loads report from file "1.fr3"
report.LoadReportFromFile("c:\1.fr3");
// Saves report to file "1.fr3"
report.SaveReportToFile("c:\2.fr3");
```

Furthermore, we will simply show only .NET notations in examples. C++ developers must remember to change examples into pointer addressing.

Designing a report

Calling the report designer is performed via the "TfrxReport.DesignReport" method. A designer opens report, which has been previously loaded by one of the LoadReport method. If no report is loaded, then designer opens an empty report.

```
report.DesignReport();
```

Running a report

Use one of the following two "TfrxReport" methods starts a report:

```
report.ShowReport();
report.PrepareReport(ClearLastReport);
```

In most cases, it is more convenient to use the first method. It displays the preview window right away, while a report continues to be constructed.
The Boolean **ClearLastReport** parameter is convenient to use in case when it is necessary to add another report to the previously constructed one (such technique is used for batch report printing).

Previewing a report

It is possible to display a report in the preview window in two ways: either by calling the "TfrxReport.ShowReport" method (described above), or with the help of the "TfrxReport.ShowPreparedReport" method. In the second case, the report construction is not performed, but a prepared report is displayed. That means that you should either construct it beforehand with the help of the "PrepareReport" method, or load a previously constructed report from the file (see chapter "Loading/saving a prepared report").

```
if (report.PrepareReport() == S_OK)
{
  frxReport1.ShowPreparedReport();
}
```

In this case, report construction is accomplished first, and after that the prepared report is displayed in the preview window. Construction of a large report can take a lot of time, and that is why it is better to use the asynchronous the "ShowReport" method. It is possible to assign preview settings by default via the "TfrxReport.PreviewOptions" property.

Printing a report

In most cases, you will print a report from the preview window. To print a report manually, you should use the "TfrxReport.Print" method, for example:

```
report.Print();
```

At the same time, the dialogue, in which printing parameters can be set, will appear. You can assign settings by default, and disable a printing dialogue with the help of the "TfrxReport.PrintOptions" property.

Loading and saving a prepared report

It can be executed from the preview window. This also can be performed manually with the help of the methods:

```
    report.SavePreparedReportToFile("PreparedReports//MyNewReport");
    report.LoadPreparedReportFromFile("PreparedReports//MyHugeReport.FP3");
```

A file, which contains the finished report, has "FP3" extension by default.

Note, that the pepared report loading is completed, its previewing is executed via the "ShowPreparedReport" method!

Exporting a report

FastReport includes powerful export abilities. The using of export slightly different for C++ and .NET managed environments.

Visual C++

To use these abilities from the C++ application code the IfrxBuiltinExports interface required. This interface should be obtained from the report object.

```
IfrxBuiltinExports* pExp;
pReport->QueryInterface(__uuidof(IfrxBuiltinExports), (void**) &pExp);
pExp->ExportToPDF( "export.pdf", true, true, true);
pExp->ExportToHTML( "export.html", true, true, true, true, true, false);
pExp->ExportToRTF( "export.rtf", true, true, true);
pExp->ExportToXLS( "export.xls", true, true, true, false, false);
pExp->ExportToXML( "export.xls", true, true, true, false);
pExp->ExportToBMP( "export.bmp", 96, true, true, true);
pExp->ExportToTIFF( "export.tif", 96, true, true, true);
pExp->ExportToJPEG( "export.jpg", 96, 50, false, true, true);
pExp->Release();
```

C#.NET, Visual Basic.NET, and others

Export in a managed environment is much easy.

```
frx.ExportToPDF( "export.pdf", true, true, true);
frx.ExportToHTML( "export.html", true, true, true, true, true, false);
frx.ExportToRTF( "export.rtf", true, true, true);
frx.ExportToXLS( "export.xls", true, true, true, false, false);
frx.ExportToXML( "export.xls", true, true, true, false);
frx.ExportToBMP( "export.bmp", 96, true, true, true);
frx.ExportToTIFF( "export.tif", 96, true, true, true);
frx.ExportToJPEG( "export.jpg", 96, 50, false, true, true);
```

Please note that export under IIS application may not be available due to disabled write access to the working directory. Therefore, you should prepare some place to export at application design time.

Now is time to describe arguments for every kind of export.

```
    HRESULT ExportToPDF(
                    [in] BSTR FileName,
                    [in] VARIANT_BOOL Compressed,
                    [in] VARIANT_BOOL EmbeddedFonts,
                    [in] VARIANT_BOOL PrintOptimized);
```

Exports prepared report to Portable Document Format.

```
    HRESULT ExportToXML(
                    [in] BSTR FileName,
                    [in] VARIANT_BOOL Styles,
                    [in] VARIANT_BOOL PageBreaks,
                    [in] VARIANT_BOOL WYSIWYG,
                    [in] VARIANT_BOOL Background);
```

Exports prepared report to XML format.

```
    HRESULT ExportToRTF(
                    [in] BSTR FileName,
                    [in] VARIANT_BOOL Pictures,
                    [in] VARIANT_BOOL PageBreaks,
                    [in] VARIANT_BOOL WYSIWYG);
```

Exports prepared report to Rich Text Format.

```
    HRESULT ExportToHTML(
                    [in] BSTR FileName,
                    [in] VARIANT_BOOL Pictures,
                    [in] VARIANT_BOOL FixedWidth,
                    [in] VARIANT_BOOL Multipage,
                    [in] VARIANT_BOOL Navigator,
                    [in] VARIANT_BOOL PicsInSameFolder,
                    [in] VARIANT_BOOL Background);
```

Exports prepared report into Portable Document Format.

```
    HRESULT ExportToXLS(
                    [in] BSTR FileName,
                    [in] VARIANT_BOOL Pictures,
                    [in] VARIANT_BOOL PageBreaks,
                    [in] VARIANT_BOOL WYSIWYG,
                    [in] VARIANT_BOOL AsText,
                    [in] VARIANT_BOOL Background);

    HRESULT ExportToBMP(
                    [in] BSTR FileName,
                    [in] int Resolution,
                    [in] VARIANT_BOOL Monochrome,
                    [in] VARIANT_BOOL CropPages,
                    [in] VARIANT_BOOL SeparatePages);

    HRESULT ExportToJPEG(
                    [in] BSTR FileName,
                    [in] int Resolution,
                    [in] int JpegQuality,
                    [in] VARIANT_BOOL Monochrome,
                    [in] VARIANT_BOOL CropPages,
                    [in] VARIANT_BOOL SeparatePages);

    HRESULT ExportToTIFF(
                    [in] BSTR FileName,
                    [in] int Resolution,
                    [in] VARIANT_BOOL Monochrome,
                    [in] VARIANT_BOOL CropPages,
                    [in] VARIANT_BOOL SeparatePages);

    HRESULT ExportToTXT([in] BSTR FileName);

    HRESULT ExportToCSV(
                    [in] BSTR FileName,
                    [in] BSTR Separator,
                    [in] VARIANT_BOOL OEMCodepage);

    HRESULT ExportToGIF(
                    [in] BSTR FileName,
                    [in] int Resolution,
                    [in] VARIANT_BOOL Monochrome,
                    [in] VARIANT_BOOL CropPages,
                    [in] VARIANT_BOOL SeparatePages);
```

```
HRESULT SendMail(
                [in] BSTR Server,
                [in] int Port,
                [in] BSTR User,
                [in] BSTR Password,
                [in] BSTR From,
                [in] BSTR Recipient,
                [in] BSTR Subject,
                [in] BSTR Text,
                [in] BSTR FileName,
                [in] BSTR AttachName);
```

Using default connection

One of abilities of the FR3 report format is storing the default connection name (alias) into report body. TfrxADOTable and TfrxADOQuery uses default ADO connection if no any TfrxADODatabase objects exist. Use standalone designer for editing default connections and assigning it to report. Use **`View->Connections`** menu item in standalone designer for edit connections. Use **`Report->Data`** menu item in standalone designer for selection connection in context of the current report.

For dynamically the assignment connection string, use following code:

```
frx.ReportOptions.ConnectionName = "MyConnectionName";
```

Note: The default connection feature is NOT CAPABLE of multithreading environments. You should create ADO connection by report designer and avoid using DafultConnection in report. If you break, this rule for multithreading environments, then access violation will happen. See the "Using in multithreading environment" issue for additional information.

Note: There is no check of availability of the ADO database by this property, so you can get error on calling `frx.PrepareReport()` method.

Note: Base connection strings are stored into registry under HKEY_LOCAL_MACHINE\SOFTWARE\Fast Reports\Connections key. This key consist of string values, where string name is a connection name and string data is the ADO connection string.

Using in multithreading environment

Using FastReport in multitasking environment did not tested well. We assume that FastReport is the pure single threading apartment model. That means what different threads can have one or several report objects, but each report object has to be accessing within parent thread context only.

Note: We found that ADO DB code is not fully multitasking compatible. Side effects raised on reports that depend on DefaultConnection property.

Building a composite report (batch printing)

In some cases it is required to organize printing of several reports at once, or capsulate and present several reports in one preview window. To perform this, there are tools in FastReport, which allow building a new report in addition to an already existing one. The «TfrxReport.PrepareReport» method has the optional «ClearLastReport» Boolean parameter, which is equal to «True» by default. This parameter defines whether it is necessary to clear pages of the previously built report. The following code shows how to build a batch from two reports:

```
frx.LoadReportFromFile("1.fr3");
frx.PrepareReport(true);
frx.LoadReportFromFile("2.fr3");
frx.PrepareReport(false);
frx.ShowPreparedReport();
```

We load the first report and build it without displaying. Then we load the second one into the same «TfrxReport» object and build it with the «ClearLastReport» parameter, equal to «false». This allows the second report to be added to the one previously built. After that, we display a finished report in the preview window.

Numbering of pages in a composite report

You can use the «Page,» «Page#,» «TotalPages,» and «TotalPages#» system variables for displaying a page number or a total number of pages. In composite reports, these variables work in the following way:

Page – page number in the current report
Page# - page number in the batch
TotalPages – total number of pages in the current report (a report must be a two-pass one)
TotalPages# - total number of pages in a batch.

Combination of pages in a composite report

As it was said above, the «PrintOnPreviousPage» property of the report design page lets you splice pages when printing, i.e. using free space of the previous page. In composite reports, it allows to start creation of a new report on free space of the previous report’s last page. To perform this, one should enable the «PrintOnPreviousPage« property of the first design page of each successive report.

Interactive reports

There are two ways to make an interactive report. The first one is using built-in script and catch events generated by dialog forms. The second one is catching event generated by report into your application.

.NET data objects

Current version of FastReport Studio provides a wrapper classes for .NET data objects. Following table shows correspondence between .NET data objects and wrappers:

.NET class	Wrapper class
DataTable	FrxDataTable
DataView	FrxDataView
DataSet	FrxDataSet

Wrapper classes ships within DataSetDemo C# project, which resides in

```
"C:\Program Files\FastReports\FastReport Studio\Demo\VisualC#.NET"
```

folder (in case of default installation path).
These wrappers provide transparent access to data by Fast Report engine. They are use TfrxUserDataSet object for getting data from .NET object and navigation on the data records. Following code fragment shows how to use data table with FastReport.

```
// Object declaration
TfrxReportClass     report;
FrxDataTable     datatable;

// Create report object
report = new TfrxReportClass();
// Create the FR compatible DataTable object
datatable = new FrxDataTable("DataTableDemo");
// add three columns to the table
datatable.Columns.Add( "id", typeof(int) );
datatable.Columns.Add( "name", typeof(string) );
datatable.Columns.Add( "onemorename", typeof(string) );
// Add ten rows to the table
for( int id=1; id<=10; id++ )
{
    datatable.Rows.Add(
        new object[] {
                    id,
                    string.Format("customer {0}", id),
                    string.Format("address {0}", 10-id) }
    );
}
// update changes
datatable.AcceptChanges();
// Load demmonstration report from file
```

```
report.LoadReportFromFile("some_report.fr3");
// Followng function binds data table to report
datatable.AssignToReport(true, report);
// You could bind data table to DataBand object (same as designer do)
datatable.AssignToDataBand("MasterData1", report);
// Show report on the screen
report.ShowReport();
```

Please, keep in mind that we cannot inherit FrxDataTable object from existing DataTable object. That means that if you receive or create the DataTable object, then you cannot bind it to report, nor convert to FrxDataTable object. To avoid this limitation we planned to implement native .NET data object support in the FastReport Studio in near future.

There is some issue on using data object with FastReport under managed environments. Due to nature of object destroying by the NET garbage collector, the object lifetime is not fully determinate. Furthermore, the FastReport engine uses global namespace of data objects within application context. In some situations, these factors can cause for unexpected results.

Actually, this is not a problem if you follow main rule: **Support unique name for each data object within an application.**

Calling external functions

FastReport Studio provides ability of calling external functions from report. As you know, FastReport shipped with built-in script engine, named FastScript, but sometime, due to some reason, you need implement some function into your code. With FastReport it is very easy - you just need to register that function and catch OnUserFunction event.

Registering function is very simple:

```
report.AddFunction(
      "function SpellValue(Value: String): String;",  // Function definition
      "User functions",                  // Function category
      "The value spelled out function");          // description
```

Catching event is slightly different for different platform. If you are using FR Studio within not managed C++ projects, in this case you need to make wrapper for IfxrReportEvens interface. We are strongly recommend use ATL IDispatchImpl for such wrapper (see Demo/VisualC++/callbacks demo). In case of managed code, we are recommends use the delegates object.
Now time to look into OnUserFunction event:

```
HRESULT OnUserFunction(
      [in] BSTR MethodName,
      [in] VARIANT Params,
      [out, retval] VARIANT* ResultValue);
```

MethodName is the name of requested function. Event handler may implement several functions, which distinguished by the MethodName argument.

Params is the parameters for the called function. The parameters shipped within VARIANT SafeArray object. That means that the fist function's argument is first the array element, second is second, and so on. The .Net framework provides System.Aray object, which is useful for fast and convenient access to the user function's arguments.

ResultValue is the result returning by the function.

Following code fragment shows how to implement UserFunction handler:

```
private object OnUserFunction(string FunctionName, object Argument)
{
   System.Array arg = (Array)Argument;
   switch(FunctionName)
   {
      case "SPELLVALUE":
         string str = (string)(arg.GetValue(0));
         return Speller.doVerbal( long.Parse(str));

      default:
         return "Undefined user function: " + FunctionName;
   }
}
```

Complete example see in UserFunctionExample C# demo, which resides in

```
"C:\Program Files\FastReports\FastReport Studio\Demo\VisualC#.NET"
```

folder (in case of default installation path).

Load text and picture from code

Sometime you may need to set text or picture on report page from application program. There is only one way to do it – use OnBeforePrint event. This event occurs for every report object every time before drawing it on report page. This event conveys one parameter – Sender with type of IfrxReportComponent. The IfrxReportComponent is base interface for every FastReport object. Therefore, you can analyze the Sender parameter and detect which object generated OnBeforePrint event.
Following example shows how to things works:

```
private void Report_OnBeforePrint(IfrxComponent Sender)
{
   // Setting text
   if(Sender.Name == "Memo2")
   {
      (Sender as IfrxCustomMemoView).Text = "This text set by application";
   }

   // Setting picture
   if (Sender.Name == "Picture1")
   {
      bmp = new Bitmap(@"..\..\2.bmp");
      IfrxPictureView  pict = (IfrxPictureView) Sender;

      pict.Picture = (int) bmp.GetHbitmap();
   }
}
```

In opposite to OnBeforePrint event, the OnAfterPrint event occurs after drawing it on report page. Therefore, we can use this event for freed resources, which was allocated in OnBeforePrint event.

```
private void Report_OnAfterPrint(IfrxComponent Sender)
{
   if (Sender is FastReport.IfrxPictureView)
   {
      bmp = null;
   }
}
```

Access report objects from a code

Sometimes you need to access to report objects, such as memos, bands, and so on. You can easily request interfaces to these objects by its names. FastReport provides two forms of FindObject method. First, one is the FindObject method of TfrxReport object. It allows requesting any object within report. Second, one is FindObject method of IfrxComponent. It allows requesting any child object, which lies behind requested object, by its name.

Visual C++

```
   IfrxComponent       *  pComponent;
   IfrxComponent       *  pMemoComp;
   IfrxMemoView        *  pMemoObj;
   IfrxReportPtr    pReport(__uuidof(TfrxReport));

   pReport->LoadReportFromFile("somereport.fr3");

   // Query base interface
   hr = pReport->QueryInterface(__uuidof(IfrxComponent), (PVOID) &pComponent);

   // Find object with name "Memo1"
   hr = pComponent->FindObject(_bstr_t("Memo1"), & pMemoComp);

   // Query memo interface from founded object
   hr = pMemoComp->QueryInterface(__uuidof(IfrxMemoView), (PVOID) & pMemoObj);

   // Set the memo text
   pMemoObj->Text = _bstr_t("This as a memo label");

   pMemoObj->Release();
   pMemoComp->Release();
   pComponent->Release();
```

C#.NET,

```
    TfrxReportClass      report;
    report.LoadReportFromFile("somereport.fr3");
```

```
// IfrxComponent is the base interface for every FastReport object
IfrxComponent        memo2;

// Find memo in report
(Report as IfrxComponent).FindObject("Memo2", out memo2);

// Assign text to memo
(memo2 as IfrxCustomMemoView).Text = "This is a new label";
```

Creating a report form from code

FastReport provides great ability for create report objects directly from program code. That means that you can dynamically create report templates without storing them on disk. For example, you can dynamically make report template based on a data structure. Object types that an allowed to create by code described in a table below:

Object name	Description
IfrxReportPage	Creates a new report page
IfrxReportTitle	Creates a report title
IfrxMemoView	Creates a memo view
IfrxReportSummary	Creates a report summary
IfrxDataBand	Creates a data band
IfrxPictureView	Creates a picture view
IfrxShapeView	Creates a shape view
IfrxChartView	Creates a chart view. Note: this feature does not available in FreeReport

Examples for dynamically report creations lie in following directories:

Language	Path do demo
C++	Examples/VisualC++/DynamicReport
C#.NET	Examples/Visual#.NET/BuiltinADO_Demo
VB.NET	Examples/VisualBasic/CreateReportByCode.VB.NET

Visual C++

```
/////////////////////////////////////////
// create instance of report object
IfrxReportPtr    pReport(__uuidof(TfrxReport));

/////////////////////////////////////////
// base interfaces
IfrxComponent    *   pComponent = NULL;
IfrxComponent    *   pReportPageComponent = NULL;

/////////////////////////////////////////
// query base interface of IfrxReport
hr = pReport->QueryInterface(__uuidof(IfrxComponent), (PVOID*) &pComponent);
if (FAILED(hr)) _com_issue_errorex(hr, pReport, __uuidof(pReport));

/////////////////////////////////////////
// create report page object
pReportPageComponent = pReport->CreateReportObject(
      pComponent,                // parent object
      __uuidof(IfrxReportPage),   // new object type
      "DynamicPage");             // new object name
```

Objects hierarchy

This topic describes the object's hierarchy of Fast Report.

IfrxComponent

The IfrxComponent is the base interface for many objects of the Fast Report. It consists of several properties and several methods.

```
interface IfrxComponent : IDispatch {

   HRESULT GetObject(
                [in] int Index,
                [out, retval] IfrxComponent** Component);
```

```
This method returns pointer to interface IfrxComponent of child object referenced by
index. Index is an integer variable in range from zero to objects count minus one.
```

```
    HRESULT BaseName([out, retval] BSTR* Value);
```

```
This property gets object base name
```

```
    HRESULT Description([out, retval] BSTR* Value);
```

```
This property gets string with object description.
```

```
    HRESULT ObjectsCount([out, retval] int* Value);
```

```
This property returns the child objects count.
```

```
    HRESULT Left([out, retval] double* Value);
```

```
This property returns the left coordinate of an object.
```

```
    HRESULT Left([in] double Value);
```

```
This property sets the left coordinate of an object.
```

```
    HRESULT Top([out, retval] double* Value);
```

```
This property returns the top coordinate of an object.
```

```
    HRESULT Top([in] double Value);
```

```
This property sets the top coordinate of an object.
```

```
    HRESULT Width([out, retval] double* Value);
```

```
This property returns an object's width.
```

```
    HRESULT Width([in] double Value);
```

```
This property sets an object's width.
```

```
    HRESULT Height([out, retval] double* Value);
```

```
This property returns an object's height.
```

```
    HRESULT Height([in] double Value);
```

This property sets an object's height.

```
    HRESULT FindObject(
                    [in] BSTR ObjectName,
                    [out, retval] IfrxComponent** Obj);
```

This method searches for object by its name. If object found thend method returns interface IfrxComponent to that object.

```
    HRESULT AliasName([out, retval] BSTR* Value);
```

This property gets an alias name of object.

```
    HRESULT Name([out, retval] BSTR* Value);
```

This property gets a name of object.

```
};
```

This chapter describes objects that are descendant to IfrxComponent interface.

IfrxReport

```
[
  odl,
  uuid(D6C97D04-9B62-450F-A6A7-B1790C39A321),
  version(1.0),
  helpstring("Dispatch interface for TfrxReport Object"),
  dual,
  oleautomation
]
interface IfrxReport : IDispatch {
    [id(0x000000cb), helpstring("Design report")]
    HRESULT DesignReport();
    [id(0x000000c9), helpstring("Show report")]
    HRESULT ShowReport();
    [id(0x000000cc), helpstring("Prepare report")]
    HRESULT PrepareReport([in] VARIANT_BOOL ClearLastReport);
    [id(0x000000cd)]
    HRESULT ShowPreparedReport();
    [id(0x000000d0)]
    HRESULT PrintReport();
    [id(0x00000077)]
    HRESULT ClearReport();
    [id(0x00000065)]
    HRESULT ExportReport([in] IfrxCustomExportFilter* Filter);
    [id(0x000000ca)]
    HRESULT LoadReportFromFile([in] BSTR szFileName);
    [id(0x00000071)]
    HRESULT SaveReportToFile([in] BSTR FileName);
    [id(0x000000ce)]
    HRESULT LoadPreparedReportFromFile([in] BSTR szFileName);
    [id(0x000000cf)]
    HRESULT SavePreparedReportToFile([in] BSTR szFileName);
    [id(0x0000006e), helpstring("Load report from stream")]
    HRESULT LoadReportFromStream([in] IUnknown* Stream);
    [id(0x0000006f), helpstring("Save report to stream")]
    HRESULT SaveReportToStream([in] IUnknown* Stream);
    [id(0x00000073), propget]
    HRESULT Errors([out, retval] BSTR* Value);
    [id(0x0000006d), propget]
    HRESULT ReportOptions([out, retval] IfrxReportOptions** Value);
    [id(0x00000070), propget]
    HRESULT PreviewOptions([out, retval] IfrxPreviewOptions** Value);
    [id(0x00000075), propget]
    HRESULT EngineOptions([out, retval] IfrxEngineOptions** Value);
    [id(0x00000076), propget]
    HRESULT PrintOptions([out, retval] IfrxPrintOptions** Value);
    [id(0x0000007b), propget]
    HRESULT ScriptLanguage([out, retval] BSTR* Value);
    [id(0x0000007b), propput]
    HRESULT ScriptLanguage([in] BSTR Value);
    [id(0x00000081), propget]
    HRESULT ScriptText([out, retval] BSTR* Value);
    [id(0x00000081), propput]
    HRESULT ScriptText([in] BSTR Value);
    [id(0x000000a0)]
    HRESULT SetVariable(
                    [in] BSTR Index,
                    [in] VARIANT Value);
    [id(0x000000a1)]
    HRESULT GetVariable(
                    [in] BSTR Index,
                    [out, retval] VARIANT* Value);
    [id(0x000000a2)]
    HRESULT AddVariable(
                    [in] BSTR Category,
                    [in] BSTR Name,
                    [in] VARIANT Value);
    [id(0x000000a3)]
    HRESULT DeleteCategory([in] BSTR Name);
    [id(0x000000a4)]
    HRESULT DeleteVariable([in] BSTR Name);
    [id(0x000000a5)]
    HRESULT SelectDataset(
                    [in] VARIANT_BOOL Selected,
                    [in] IUnknown* DataSet);
    [id(0x00000072)]
    HRESULT CreateReportObject(
                    [in] IfrxComponent* ParentObject,
                    [in] GUID ObjectType,
                    [in] BSTR Name,
                    [out, retval] IfrxComponent** GeneratedObject);
    [id(0x00000066)]
    HRESULT LoadLanguageResourcesFromFile([in] BSTR FileName);
    [id(0x00000067)]
    HRESULT GetResourceString(
                    [in] BSTR ID,
                    [out, retval] BSTR* ResourceStr);
    [id(0x00000068), propput]
    HRESULT MainWindowHandle([in] long rhs);
    [id(0x00000069), propput]
    HRESULT ShowProgress([in] VARIANT_BOOL rhs);
    [id(0x0000006a)]
    HRESULT AddFunction(
                    [in] BSTR FunctionName,
                    [in] BSTR Category,
                    [in] BSTR Description);
    [id(0x0000006b)]
    HRESULT SavePreparedReportToStream([in] IUnknown* Stream);
    [id(0x0000006c), propget]
    HRESULT Resources([out, retval] IfrxResources** Value);
    [id(0x00000074), propget]
    HRESULT Version([out, retval] BSTR* Value);
    [id(0x00000078)]
    HRESULT BindObject([in] IUnknown* Value);
    [id(0x00000079)]
    HRESULT Get_Page(
                    [in] long Index,
                    [out, retval] IfrxPage** Value);
    [id(0x0000007a), propget]
    HRESULT PagesCount([out, retval] long* Value);
    [id(0x000000d1)]
    HRESULT CreateReportObjectEx(
                    [in] IfrxComponent* ParentObject,
                    [in] BSTR ObjectType,
                    [in] BSTR Name,
                    [out, retval] IfrxComponent** GeneratedObject);
    [id(0x000000d2), helpstring("Search for object by his name. Returns  IfrxComponent
interface for founded object.")]
    HRESULT FindObject(
                    [in] BSTR ObjectName,
                    [out, retval] IfrxComponent** Obj);
};
```

IfrxDataSet

```
[
  odl,
  uuid(AF1529F7-431B-4105-B222-185180B1CAD5),
  version(1.0),
  helpstring("Base dataset object"),
  dual,
  oleautomation
]
interface IfrxDataSet : IDispatch {
    [id(0x00000065), propget]
    HRESULT UserName([out, retval] BSTR* Value);
    [id(0x00000065), propput]
    HRESULT UserName([in] BSTR Value);
    [id(0x00000066), propget]
    HRESULT RangeBegin([out, retval] frxRangeBegin* Value);
    [id(0x00000066), propput]
    HRESULT RangeBegin([in] frxRangeBegin Value);
    [id(0x00000067), propget]
    HRESULT RangeEndCount([out, retval] int* Value);
    [id(0x00000067), propput]
    HRESULT RangeEndCount([in] int Value);
    [id(0x00000068), propget]
    HRESULT RangeEnd([out, retval] frxRangeEnd* Value);
    [id(0x00000068), propput]
    HRESULT RangeEnd([in] frxRangeEnd Value);
};
```

IfrxUserDataSet

```
[
  odl,
  uuid(0ED40089-6575-4110-B24D-B09DD08C6F73),
  version(1.0),
  helpstring("Aggregate some of methods and properties of the TfrxUserDataSet class"),
  oleautomation
]
interface IfrxUserDataSet : IDispatch {
    [propget, helpstring("EndOfLine terminated list of UserDataSet fields ")]
    HRESULT _stdcall Fields([out, retval] BSTR* Value);
    [propput, helpstring("EndOfLine terminated list of UserDataSet fields ")]
    HRESULT _stdcall Fields([in] BSTR Value);
    [propget, helpstring("The DataSet name (UserName)")]
    HRESULT _stdcall Name([out, retval] BSTR* Value);
    [propput, helpstring("The DataSet name (UserName)")]
    HRESULT _stdcall Name([in] BSTR Value);
};
```

IfrxADOTable

```
[
  odl,
  uuid(69999585-A29C-4BEC-8102-A370F2178B06),
  version(1.0),
  oleautomation
]
interface IfrxADOTable : IDispatch {
    [propget]
    HRESULT _stdcall DataBase([out, retval] IfrxADODatabase** Value);
    [propput]
    HRESULT _stdcall DataBase([in] IfrxADODatabase* Value);
    [propget]
    HRESULT _stdcall IndexName([out, retval] BSTR* Value);
    [propput]
    HRESULT _stdcall IndexName([in] BSTR Value);
    [propget]
    HRESULT _stdcall TableName([out, retval] BSTR* Value);
    [propput]
    HRESULT _stdcall TableName([in] BSTR Value);
    [propget]
    HRESULT _stdcall Name([out, retval] BSTR* Value);
    [propput]
    HRESULT _stdcall Name([in] BSTR Value);
};
```

IfrxADOQuery

```
[
  odl,
  uuid(6ABB2DE0-8424-4529-A246-5B263B41CEBA),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxADOQuery : IDispatch {
    [id(0x00000065), propget]
    HRESULT DataBase([out, retval] IfrxADODatabase** Value);
    [id(0x00000065), propput]
    HRESULT DataBase([in] IfrxADODatabase* Value);
    [id(0x00000066), propget]
    HRESULT Query([out, retval] BSTR* Value);
    [id(0x00000066), propput]
    HRESULT Query([in] BSTR Value);
    [id(0x00000067), propget]
    HRESULT Name([out, retval] BSTR* Value);
    [id(0x00000067), propput]
    HRESULT Name([in] BSTR Value);
    [id(0x00000068)]
    HRESULT ParamByName(
                    [in] BSTR Name,
                    [out, retval] IfrxParamItem** Param);
};
```

IfrxADODatabase

```
[
  odl,
  uuid(2E333A88-6D94-45AB-85A3-3D7AA3BAA83B),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxADODatabase : IDispatch {
    [id(0x00000065), propget]
    HRESULT ConnectionString([out, retval] BSTR* Value);
    [id(0x00000065), propput]
    HRESULT ConnectionString([in] BSTR Value);
    [id(0x00000066), propget]
    HRESULT LoginPrompt([out, retval] VARIANT_BOOL* Value);
    [id(0x00000066), propput]
    HRESULT LoginPrompt([in] VARIANT_BOOL Value);
    [id(0x00000067), propget]
    HRESULT Connected([out, retval] VARIANT_BOOL* Value);
    [id(0x00000067), propput]
    HRESULT Connected([in] VARIANT_BOOL Value);
};
```

IfrxPage

```
[
  odl,
  uuid(37585ADA-EE55-4E9D-AF49-E178E61D3BAE),
  version(1.0),
  oleautomation
]
interface IfrxPage : IUnknown {
    [propget]
    HRESULT _stdcall Visible([out, retval] VARIANT_BOOL* Value);
    [propput]
    HRESULT _stdcall Visible([in] VARIANT_BOOL Value);
};
```

IfrxReportPage

```
[
  odl,
  uuid(10029A3D-D7CB-449A-90E7-3FA255F50E39),
  version(1.0),
  dual,
  oleautomation
```

```
]
interface IfrxReportPage : IDispatch {
    [id(0x00000066)]
    HRESULT SetDefaults();
    [id(0x00000065), propget]
    HRESULT Bin([out, retval] int* Value);
    [id(0x00000065), propput]
    HRESULT Bin([in] int Value);
    [id(0x00000067), propget]
    HRESULT BinOtherPages([out, retval] int* Value);
    [id(0x00000067), propput]
    HRESULT BinOtherPages([in] int Value);
    [id(0x00000068), propget]
    HRESULT BottomMargin([out, retval] double* Value);
    [id(0x00000068), propput]
    HRESULT BottomMargin([in] double Value);
    [id(0x00000069), propget]
    HRESULT Columns([out, retval] int* Value);
    [id(0x00000069), propput]
    HRESULT Columns([in] int Value);
    [id(0x0000006a), propget]
    HRESULT ColumnWidth([out, retval] double* Value);
    [id(0x0000006a), propput]
    HRESULT ColumnWidth([in] double Value);
    [id(0x0000006b), propget]
    HRESULT ColumnPositions([out, retval] BSTR* Value);
    [id(0x0000006b), propput]
    HRESULT ColumnPositions([in] BSTR Value);
    [id(0x0000006c), propget]
    HRESULT DataSet([out, retval] IfrxDataSet** Value);
    [id(0x0000006c), propput]
    HRESULT DataSet([in] IfrxDataSet* Value);
    [id(0x0000006d), propget]
    HRESULT Duplex([out, retval] frxDuplexMode* Value);
    [id(0x0000006d), propput]
    HRESULT Duplex([in] frxDuplexMode Value);
    [id(0x0000006e), propget]
    HRESULT HGuides([out, retval] BSTR* Value);
    [id(0x0000006e), propput]
    HRESULT HGuides([in] BSTR Value);
    [id(0x0000006f), propget]
    HRESULT LargeDesignHeight([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006f), propput]
    HRESULT LargeDesignHeight([in] VARIANT_BOOL Value);
    [id(0x00000070), propget]
    HRESULT LeftMargin([out, retval] double* Value);
    [id(0x00000070), propput]
    HRESULT LeftMargin([in] double Value);
    [id(0x00000071), propget]
    HRESULT MirrorMargins([out, retval] VARIANT_BOOL* Value);
    [id(0x00000071), propput]
    HRESULT MirrorMargins([in] VARIANT_BOOL Value);
    [id(0x00000072), propget]
    HRESULT Orientation([out, retval] PrinterOrientation* Value);
    [id(0x00000072), propput]
    HRESULT Orientation([in] PrinterOrientation Value);
    [id(0x00000073), propget]
    HRESULT OutlineText([out, retval] BSTR* Value);
    [id(0x00000073), propput]
    HRESULT OutlineText([in] BSTR Value);
    [id(0x00000074), propget]
    HRESULT PrintIfEmpty([out, retval] VARIANT_BOOL* Value);
    [id(0x00000074), propput]
    HRESULT PrintIfEmpty([in] VARIANT_BOOL Value);
    [id(0x00000075), propget]
    HRESULT PrintOnPreviousPage([out, retval] VARIANT_BOOL* Value);
    [id(0x00000075), propput]
    HRESULT PrintOnPreviousPage([in] VARIANT_BOOL Value);
    [id(0x00000076), propget]
    HRESULT RightMargin([out, retval] double* Value);
    [id(0x00000076), propput]
    HRESULT RightMargin([in] double Value);
    [id(0x00000077), propget]
    HRESULT SubReport([out, retval] IUnknown** Value);
    [id(0x00000077), propput]
    HRESULT SubReport([in] IUnknown* Value);
    [id(0x00000078), propget]
    HRESULT TitleBeforeHeader([out, retval] VARIANT_BOOL* Value);
    [id(0x00000078), propput]
    HRESULT TitleBeforeHeader([in] VARIANT_BOOL Value);
    [id(0x00000079), propget]
    HRESULT TopMargin([out, retval] double* Value);
    [id(0x00000079), propput]
    HRESULT TopMargin([in] double Value);
    [id(0x0000007a), propget]
    HRESULT VGuides([out, retval] BSTR* Value);
    [id(0x0000007a), propput]
    HRESULT VGuides([in] BSTR Value);
    [id(0x0000007b), propget]
    HRESULT BackPickture([out, retval] IUnknown** Value);
    [id(0x0000007b), propput]
    HRESULT BackPickture([in] IUnknown* Value);
};
```

IfrxView

```
[
  odl,
  uuid(068E07A4-8AA5-41A0-B9DF-12CAB39A964D),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxView : IDispatch {
    [id(0x00000065), propget]
    HRESULT DataField([out, retval] BSTR* Value);
    [id(0x00000065), propput]
    HRESULT DataField([in] BSTR Value);
    [id(0x00000067), propget]
    HRESULT TagStr([out, retval] BSTR* Value);
    [id(0x00000067), propput]
    HRESULT TagStr([in] BSTR Value);
    [id(0x00000068), propget]
    HRESULT URL([out, retval] BSTR* Value);
    [id(0x00000068), propput]
    HRESULT URL([in] BSTR Value);
    [id(0x00000066), propget]
    HRESULT DataSetName([out, retval] BSTR* Value);
    [id(0x00000066), propput]
    HRESULT DataSetName([in] BSTR Value);
    [id(0x00000069), propget]
    HRESULT Name([out, retval] BSTR* Value);
};
```

IfrxCustomMemoView

```
[
  odl,
  uuid(189CC3B5-EAC5-4652-98C7-A83472B73592),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxCustomMemoView : IDispatch {
    [id(0x00000065), propget]
    HRESULT Text([out, retval] BSTR* Value);
    [id(0x00000065), propput]
    HRESULT Text([in] BSTR Value);
};
```

IfrxMemoView

```
[
  odl,
  uuid(EFDC52E3-63D4-475A-93DD-6ACC8CEEA7FC),
  version(1.0),
  helpstring("The MemoView interface. It inherits a?o properties of the parent
objects"),
  dual,
  oleautomation
]
interface IfrxMemoView : IDispatch {
    [id(0x00000065), propget]
    HRESULT AutoWidth([out, retval] VARIANT_BOOL* Value);
    [id(0x00000065), propput]
    HRESULT AutoWidth([in] VARIANT_BOOL Value);
    [id(0x00000066), propget]
    HRESULT AllowExpressions([out, retval] VARIANT_BOOL* Value);
    [id(0x00000066), propput]
    HRESULT AllowExpressions([in] VARIANT_BOOL Value);
    [id(0x00000067), propget]
    HRESULT AllowHTMLTags([out, retval] VARIANT_BOOL* Value);
    [id(0x00000067), propput]
    HRESULT AllowHTMLTags([in] VARIANT_BOOL Value);
    [id(0x00000068), propget]
    HRESULT BrushStyle([out, retval] TfrxBrushStyle* Value);
    [id(0x00000068), propput]
    HRESULT BrushStyle([in] TfrxBrushStyle Value);
    [id(0x00000069), propget]
    HRESULT CharSpacing([out, retval] double* Value);
    [id(0x00000069), propput]
    HRESULT CharSpacing([in] double Value);
    [id(0x0000006a), propget]
    HRESULT Clipped([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006a), propput]
    HRESULT Clipped([in] VARIANT_BOOL Value);
    [id(0x0000006b), propget]
    HRESULT Color([out, retval] long* Value);
    [id(0x0000006b), propput]
    HRESULT Color([in] long Value);
    [id(0x0000006c), propget]
    HRESULT DataField([out, retval] BSTR* Value);
    [id(0x0000006c), propput]
    HRESULT DataField([in] BSTR Value);
    [id(0x0000006d), propget]
    HRESULT DataSet([out, retval] IfrxDataSet** Value);
    [id(0x0000006d), propput]
    HRESULT DataSet([in] IfrxDataSet* Value);
    [id(0x0000006e), propget]
    HRESULT DataSetName([out, retval] BSTR* Value);
    [id(0x0000006e), propput]
    HRESULT DataSetName([in] BSTR Value);
    [id(0x0000006f), propget]
    HRESULT DisplayFormat([out, retval] IfrxDisplayFormat** Value);
    [id(0x00000070), propget]
    HRESULT ExpressionDelimiters([out, retval] BSTR* Value);
    [id(0x00000070), propput]
    HRESULT ExpressionDelimiters([in] BSTR Value);
    [id(0x00000071), propget]
    HRESULT FlowTo([out, retval] IfrxCustomMemoView** Value);
    [id(0x00000071), propput]
    HRESULT FlowTo([in] IfrxCustomMemoView* Value);
    [id(0x00000072), propget]
    HRESULT Font([out, retval] IfrxFont** Value);
    [id(0x00000073), propget]
    HRESULT Frame([out, retval] IfrxFrame** Value);
    [id(0x00000074), propget]
    HRESULT GapX([out, retval] double* Value);
    [id(0x00000074), propput]
    HRESULT GapX([in] double Value);
    [id(0x00000075), propget]
    HRESULT GapY([out, retval] double* Value);
    [id(0x00000075), propput]
    HRESULT GapY([in] double Value);
    [id(0x00000076), propget]
    HRESULT HAlign([out, retval] frxHAlign* Value);
    [id(0x00000076), propput]
    HRESULT HAlign([in] frxHAlign Value);
    [id(0x00000077), propget]
    HRESULT HideZeros([out, retval] VARIANT_BOOL* Value);
    [id(0x00000077), propput]
    HRESULT HideZeros([in] VARIANT_BOOL Value);
    [id(0x00000078), propget]
    HRESULT Highlight([out, retval] IfrxHighlight** Value);
    [id(0x00000079), propget]
    HRESULT LineSpacing([out, retval] double* Value);
    [id(0x00000079), propput]
    HRESULT LineSpacing([in] double Value);
    [id(0x0000007a), propget]
    HRESULT Memo([out, retval] BSTR* Value);
    [id(0x0000007a), propput]
    HRESULT Memo([in] BSTR Value);
    [id(0x0000007b), propget]
    HRESULT ParagraphGap([out, retval] double* Value);
    [id(0x0000007b), propput]
    HRESULT ParagraphGap([in] double Value);
    [id(0x0000007c), propget]
    HRESULT ParentFont([out, retval] VARIANT_BOOL* Value);
    [id(0x0000007c), propput]
    HRESULT ParentFont([in] VARIANT_BOOL Value);
    [id(0x0000007d), propget]
    HRESULT Rotation([out, retval] long* Value);
    [id(0x0000007d), propput]
    HRESULT Rotation([in] long Value);
    [id(0x0000007e), propget]
    HRESULT RTLReading([out, retval] VARIANT_BOOL* Value);
    [id(0x0000007e), propput]
    HRESULT RTLReading([in] VARIANT_BOOL Value);
    [id(0x0000007f), propget]
    HRESULT Style([out, retval] BSTR* Value);
    [id(0x0000007f), propput]
    HRESULT Style([in] BSTR Value);
    [id(0x00000080), propget]
    HRESULT SuppressRepeated([out, retval] VARIANT_BOOL* Value);
    [id(0x00000080), propput]
    HRESULT SuppressRepeated([in] VARIANT_BOOL Value);
    [id(0x00000081), propget]
    HRESULT Underlines([out, retval] VARIANT_BOOL* Value);
    [id(0x00000081), propput]
    HRESULT Underlines([in] VARIANT_BOOL Value);
    [id(0x00000082), propget]
    HRESULT WordBreak([out, retval] VARIANT_BOOL* Value);
    [id(0x00000082), propput]
    HRESULT WordBreak([in] VARIANT_BOOL Value);
    [id(0x00000083), propget]
    HRESULT WordWrap([out, retval] VARIANT_BOOL* Value);
    [id(0x00000083), propput]
    HRESULT WordWrap([in] VARIANT_BOOL Value);
    [id(0x00000084), propget]
    HRESULT VAlign([out, retval] frxVAlign* Value);
    [id(0x00000084), propput]
    HRESULT VAlign([in] frxVAlign Value);
};
```

IfrxPictureView

```
[
  odl,
  uuid(681EFCCC-AE92-4092-A40A-42EED8E8AE79),
  version(1.0),
  helpstring("This interface reprents FastReport picture"),
  dual,
  oleautomation
]
interface IfrxPictureView : IDispatch {
    [id(0x00000065), propget]
    HRESULT Picture([out, retval] OLE_HANDLE* Value);
    [id(0x00000065), propput]
    HRESULT Picture([in] OLE_HANDLE Value);
    [id(0x00000066)]
    HRESULT LoadViewFromStream([in] IUnknown* Stream);
    [id(0x00000067)]
    HRESULT SaveViewToStream([in] IUnknown* Stream);
};
```

IfrxChartView

```
[
  odl,
  uuid(7AE012B7-B452-4110-8CDC-3AC928DA7015),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxChartView : IDispatch {
    [id(0x00000065)]
    HRESULT GetSeriesItem(
                    [in] long Index,
                    [out, retval] IfrxSeriesItem** Value);
    [id(0x00000066)]
    HRESULT AddSeriesItem([out, retval] IfrxSeriesItem** NewItem);
};
```

IfrxShapeView

```
[
  odl,
  uuid(1DF07E4C-6E6A-4AFC-954A-A9B2B91A9D94),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxShapeView : IDispatch {
    [id(0x00000065), propget]
    HRESULT Curve([out, retval] long* Value);
    [id(0x00000065), propput]
    HRESULT Curve([in] long Value);
    [id(0x00000066), propget]
    HRESULT ShapeType([out, retval] frxShapeType* Value);
    [id(0x00000066), propput]
    HRESULT ShapeType([in] frxShapeType Value);
};
```

IfrxOLEView

```
[
  odl,
  uuid(FAA41972-B0B1-4E70-BEA9-F9373C0B6F1A),
  version(1.0),
  dual,
  oleautomation
]
interface IfrxOLEView : IDispatch {
    [id(0x00000065), propget]
    HRESULT OleContainer([out, retval] IUnknown** Value);
    [id(0x00000066), propget]
    HRESULT SizeMode([out, retval] long* Value);
    [id(0x00000066), propput]
    HRESULT SizeMode([in] long Value);
    [id(0x00000067), propget]
    HRESULT Stretched([out, retval] VARIANT_BOOL* Value);
    [id(0x00000067), propput]
    HRESULT Stretched([in] VARIANT_BOOL Value);
};
```

IfrxRichView

```
[
  odl,
  uuid(B78C53E0-2640-44A9-B796-40D54488DDED),
  version(1.0),
  helpstring("Rich view interface"),
  dual,
  oleautomation
]
interface IfrxRichView : IDispatch {
    [id(0x00000065)]
    HRESULT LoadViewFromStream([in] IUnknown* Stream);
    [id(0x00000066)]
    HRESULT SaveViewToStream([in] IUnknown* Stream);
};
```

IfrxBand

```
[
  odl,
  uuid(1329B776-F91F-4607-84BE-BE8B166C104D),
  version(1.0),
  helpstring("Properties of TfrxBandInterface"),
  dual,
  oleautomation
]
interface IfrxBand : IDispatch {
    [id(0x00000065), propget]
    HRESULT AllowSplit([out, retval] VARIANT_BOOL* Value);
    [id(0x00000065), propput]
    HRESULT AllowSplit([in] VARIANT_BOOL Value);
    [id(0x00000066), propget]
    HRESULT KeepChild([out, retval] VARIANT_BOOL* Value);
    [id(0x00000066), propput]
    HRESULT KeepChild([in] VARIANT_BOOL Value);
    [id(0x000000c9), propget]
    HRESULT OutlineText([out, retval] BSTR* Value);
    [id(0x000000c9), propput]
    HRESULT OutlineText([in] BSTR Value);
    [id(0x000000ca), propget]
    HRESULT Overflow([out, retval] VARIANT_BOOL* Value);
    [id(0x000000ca), propput]
    HRESULT Overflow([in] VARIANT_BOOL Value);
    [id(0x000000cb), propget]
    HRESULT StartNewPage([out, retval] VARIANT_BOOL* Value);
    [id(0x000000cb), propput]
    HRESULT StartNewPage([in] VARIANT_BOOL Value);
    [id(0x000000cc), propget]
    HRESULT Stretched([out, retval] VARIANT_BOOL* Value);
    [id(0x000000cc), propput]
    HRESULT Stretched([in] VARIANT_BOOL Value);
    [id(0x000000cd), propget]
    HRESULT PrintChildIfInvisible([out, retval] VARIANT_BOOL* Value);
    [id(0x000000cd), propput]
    HRESULT PrintChildIfInvisible([in] VARIANT_BOOL Value);
    [id(0x000000ce), propget]
    HRESULT Vertical([out, retval] VARIANT_BOOL* Value);
    [id(0x000000ce), propput]
    HRESULT Vertical([in] VARIANT_BOOL Value);
    [id(0x000000cf), propget]
    HRESULT BandName([out, retval] BSTR* Value);
};
```

IfrxDataBand

```
[
  odl,
  uuid(2E24ACD0-5C03-43DC-8417-0371133C1609),
  version(1.0),
  dual,
  oleautomation
]
```

```
interface IfrxDataBand : IDispatch {
    [id(0x00000065), propget]
    HRESULT ColumnGap([out, retval] double* Value);
    [id(0x00000065), propput]
    HRESULT ColumnGap([in] double Value);
    [id(0x00000066), propget]
    HRESULT ColumnWidth([out, retval] double* Value);
    [id(0x00000066), propput]
    HRESULT ColumnWidth([in] double Value);
    [id(0x00000067), propget]
    HRESULT ColumnsCount([out, retval] long* Value);
    [id(0x00000067), propput]
    HRESULT ColumnsCount([in] long Value);
    [id(0x00000068), propget]
    HRESULT CurrentColumn([out, retval] long* Value);
    [id(0x00000068), propput]
    HRESULT CurrentColumn([in] long Value);
    [id(0x00000069), propget]
    HRESULT DataSet([out, retval] IfrxDataSet** Value);
    [id(0x00000069), propput]
    HRESULT DataSet([in] IfrxDataSet* Value);
    [id(0x0000006a), propget]
    HRESULT FooterAfterEach([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006a), propput]
    HRESULT FooterAfterEach([in] VARIANT_BOOL Value);
    [id(0x0000006b), propget]
    HRESULT KeepFooter([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006b), propput]
    HRESULT KeepFooter([in] VARIANT_BOOL Value);
    [id(0x0000006c), propget]
    HRESULT KeepHeader([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006c), propput]
    HRESULT KeepHeader([in] VARIANT_BOOL Value);
    [id(0x0000006d), propget]
    HRESULT KeepTogether([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006d), propput]
    HRESULT KeepTogether([in] VARIANT_BOOL Value);
    [id(0x0000006e), propget]
    HRESULT PrintIfDetailEmpty([out, retval] VARIANT_BOOL* Value);
    [id(0x0000006e), propput]
    HRESULT PrintIfDetailEmpty([in] VARIANT_BOOL Value);
    [id(0x0000006f), propget]
    HRESULT RowCount([out, retval] long* Value);
    [id(0x0000006f), propput]
    HRESULT RowCount([in] long Value);
    [id(0x00000070), helpstring("Unlink DataSet from DataBand")]
    HRESULT ResetDataSet();
};
```

IfrxReportTitle

```
[
  odl,
  uuid(4D20B8BF-CA72-4FB5-A1BF-A3C542650910),
  version(1.0),
  helpstring("used for CreateReportObject() function"),
  dual,
  oleautomation
]
interface IfrxReportTitle : IDispatch {
};
```

IfrxReportSummary

```
[
  odl,
  uuid(634AD4C9-74CF-4F08-B453-FD8E1E411075),
  version(1.0),
  helpstring("TfrxReportSummary"),
  oleautomation
]
interface IfrxReportSummary : IUnknown {
};
```

Deploying Applications

Deployment is the process by which you distribute a finished application to be installed on other computers.

Before you run your application on another computer, which have no FR Studio installed, you should copy the FastReport dynamic library and language resource file[s] on the that computer.

Default installation path:	`"C:\Program Files\FastReports\FastReport Studio\Bin\"`
DLL name:	`FasrReport.dll`
Language resources files	`English.frc, Russian.frc, etc`

In addition, you should register dynamic library on target computer by following command at the shell prompt:

```
regsvr32.exe FastReport.dll
```

Finally, you need set the FastReport language resource in the target computer registry:

```
[HKEY_CURRENT_USER\Software\Fast Reports\Resources]
"Language"="English"
```

Binary compatibility issues

We are not warranty the binary compatibility between releases. Due to rapid development of FastReport Studio, from time to time we are have changing interfaces (mostly append new methods and properties). So, from time to time interface changing. As result, you application can works in improper way or raise access violation. To avoid such behavior you should recompile your projects, which depends on FastReport Studio. In most cases, just recompile is enough, but sometime you need make changes in code to fit latest version of FastReport Studio. We are sorry for inconvenience.