FastReport® Studio

Programmer's manual

Edition 1.09

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
1.09	June 5, 2006	ADO NET wrappers update TfrxGzipCompressor description added Described the SendMail method of IfrxBuiltin export
1.10	June 9, 2006	Added "Auxilary interfaces" topic
1.11	June 145, 2006	Added description for all bands

Table of Contents

Foreword

This documentation consists of several related chapters that describe different aspects of using FastReport.

"*FastReport object review*" chapter describes FastReport Studio co-classes. These classes may be created directly by application without using another FR Studio classes or interfaces. In addition, this chapter gives description of interfaces and disipinterfaces that are not descendant of IfrxComponent, so these interfaces are not matches to objects hierarchy.

"*Working with TfrxReport object*" chapter is analogue of Quick Start guide. This chapter gives some examples, advices and recommendations how to accomplish simple functions. It includes examples for C++, C#, and VB.NET emvironments. Generally, most examples are shown for managed and unmanaged code.

"*Objects hierarchy*" chapter provides an object's inheritances relation information. If you are not familiar with original Fast Report Delphi, then it is only place to get information about interfaces relation. In other words, this chapter describes all interfaces that are successors of IfrxComponent interface.

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. This chapter describes FastReport co-classes.

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 of TfrxReport co-class:

```
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 into category.

```
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. ObjectType argument is a GUID. The list of

allowed GUIDs described in "Working with TfrxReport" chapter "Creating a report from code" topic.

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

This method is similar to CreatereportObject except that ObjectType argument represented by string. This method introduced to make possible object creation under Visual Basic 6.

```
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.

```
HRESULT _stdcall SavePreparedReportToStream([in] IUnknown* Stream);
```

This method saves prepared report into COM or .NET stream. Method can be used for saving prepared report into database or sending prepared report over network, or sending prepared report into any protocol or storgae that support streams.

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

This method gets FR Studio version as a string value. It obtains only major and minor versions but not a build number.

```
HRESULT _stdcall Get_Page(
                    [in] long Index,
                    [out, retval] IfrxPage** Value);
```

Get report page object by its number. Return value has an IfrxPage type.

```
HRESULT _stdcall PagesCount([out, retval] long* Value);
```

Get report's pages count. This value is a template pages count, not prepared pages.

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

Searches report for object with a given ObjectName and return its IfrxComponent interface. IfrxComponent interface is described in the "Objects hierarchy" chapter. Note that the ObjectName is a case sensitive string.

```
HRESULT _stdcall PreviewPages([out, retval] IfrxCustomPreviewPages** Value);
```

Returns interface on the preview pages storage. This storage holds all prepared pages that are generated by PrepareReport method. See description on IfrxCustomPreviewPages interface below.

```
HRESULT _stdcall ClearDatasets();
```

```
Resets all datasest that are binded to report.
```

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);
```

Gives 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.

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

This property is useful for batch-based environments and it disables report built-in dialogs.

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-".
This property affects to the exports too.

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 template:

```
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.

};
```

Interface that controls the storage that holds all prepared pages that are generated by PrepareReport method. Use this interface carefully, because it provides an internal low-level methods and properties.

```
interface IfrxCustomPreviewPages : IDispatch {

    HRESULT _stdcall AddObject([in] IfrxComponent* Value);

An internal method. Must be not used at all.

    HRESULT _stdcall AddPage([in] IfrxReportPage* Value);

Add page to the pool of prepared pages.

    HRESULT _stdcall AddEmptyPage([in] long Index);

Add an emty page with index that set by the 'Index' argument.

    HRESULT _stdcall DeletePage([in] long Index);

Delete page with idex that set by the 'Index' argument

    HRESULT _stdcall Count([out, retval] long* Value);
```

Get count of prepared pages.

```
    HRESULT _stdcall CurrentPage([out, retval] long* Value);
```

Get index of current page.

```
    HRESULT _stdcall CurrentPage([in] long Value);
```

Set current page by Index.

```
    HRESULT _stdcall CurPreviewPage([out, retval] long* Value);
    HRESULT _stdcall CurPreviewPage([in] long Value);
```

Get/set page that currently displayed in preview window

```
    HRESULT _stdcall Page(
                    [in] long Index,
                    [out, retval] IfrxReportPage** Value);
```

Get page by its index. Return value is an page described by IfrxReportPage interface. This interface described in 'Objects hierarchy' chapter

```
};
```

Events of the TfrxReport object

There are several events are defined in the TfrxReport object. Most of events have a 'sender' argument that points to the object, which the event source. The 'sender' argument is a base object's interface. It is possible to detect an object type by using BaseName property.

```
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). This object also used in ADO NET wrapper classes to provide communication layer between ADO NET objects and FastReport objects. This object does not have any methods, but have properties and events.

Note for ADO NET programmers: FastReport terminology is slightly differs from ADO NET terminology. In terms of ADO NET the DataSet consist of number of Tables, Views and Relations, but FastReport dataset is analogue to the ADO NET table. Please do not be confused with terminology.

Properties

The TfrxUserDataSet component has the following properties:

```
interface IfrxUserDataSet : IDispatch
```

```
{

   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.

```
};
```

This object is inherited from TfrxDataSet, see Objects hierarchy issue for the description IfrxDataSet.

Events

The following events are defined for the TfrxUserDataSet object. These events must be handled by application program to provide data to the report.

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

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

```
   HRESULT _stdcall 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 _stdcall OnFirst();
```

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

```
   HRESULT _stdcall OnNext();
```

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

```
   HRESULT _stdcall OnPrior();
```

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

```
};
```

Builtin 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".

There are two ways for ADO objects creation. Generally, ADO objects are embedded into report, so it is need to request interfaces to these objects after LoadReport by means of **FindObject** method of IfrxReport interface.

Anohter way of creation ADO object is creation them by application. In this case you need to assign TfrxADOTable and TfrxADOQuery to report by means of **SelectDataset** method of IfrxReport interface. In addition, you may need to assign these objects to report band(s) by means of **DataSet** property of IfrxDataBand interface.

TfrxADODatabase object

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

```
interface IfrxADODatabase : IDispatch {

   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 or 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 this property to determine the current status of a connection. If Connected is true, the connection is active; if false, then the connection is inactive.

```
   HRESULT  _stdcall ConnectionTimeout([out, retval] long* Value);
   HRESULT  _stdcall ConnectionTimeout([in] long Value);
```

Specifies a timeout value for a database connection procedure.

```
};
```

TfrxADOTable object

TfrxADOTable is a dataset component that encapsulates a table accessed through an ADO data store. This table used as a data source for report building. This object's default interface is a descendant IfrxDataSet interface.

```
interface IfrxADOTable : IDispatch {

   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. This object's default interface is a descendant IfrxDataSet interface.

```
interface IfrxADOQuery : IDispatch {

   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.

```
    HRESULT _stdcall CommandTimeout([out, retval] long* Value);
    HRESULT _stdcall CommandTimeout([in] long Value);
```

Specifies timeout value for a SQL statement. In some rare cases, you may need to increase this value to allow execute long time processing query.

```
    HRESULT _stdcall ParamByName(
        [in] BSTR Name,
        [out, retval] IfrxParamItem** Param);

};
```

ActiveX objects

FR Studio includes two ActiveX objects – preview window and preview form. The difference between them is a user graphical control set. Preview window displays only a report preview window without any controls, and preview form is a form, which includes the preview window and some control objects such as zoom, exports, navigation buttons, and other controls.

TfrxPreviewX

TfrxPreviewX ActiveX control provides the means to display preview window to display prepared reports. It does not include any additional user interface controls. TfrxPreviewX object implements two interfaces IfrxPreview and IfrxPreview events.

```
dispinterface IfrxPreview {

        void _stdcall Init();
        void _stdcall Lock();
        void _stdcall Unlock();
```

Internal ActiveX methods

```
        void _stdcall AddPage();
```

Add single empty page to prepared report.

```
        void _stdcall DeletePage();
```

Deletes current page in prepared report.

```
        void _stdcall Print();
```

Prints prepared report on selected printer.

```
        void _stdcall LoadFromFile([in] BSTR FileName);
```

Loads prepared report from a file, which name set by FileName argument.

```
        void _stdcall SaveToFile([in] BSTR FileName);
```

Saves prepared report to a file, which name set by FileName argument.

```
        void _stdcall Edit();
```

Run editor on prepared report.

```
        void _stdcall First();
        void _stdcall Next();
        void _stdcall Prior();
        void _stdcall Last();
```

Provides navigation within prepared report.

```
        void _stdcall PageSetupDlg();
```

Displays the PageSetup dialog

```
        void _stdcall Find();
        void _stdcall FindNext();
```

Displays the Find dialog

```
        void _stdcall Clear();
```

Clear prepared report.

```
        void _stdcall SetPosition(
                        [in] long PageN,
                        [in] long Top);
```

Set current position

```
        void _stdcall ShowMessage([in] BSTR s);
        void _stdcall HideMessage();
```

Display/hide window on top of preview window

```
        void _stdcall MouseWheelScroll(
                        [in] long Delta,
                        [in] VARIANT_BOOL Horz,
                        [in] VARIANT_BOOL Zoom);
```

Emulates mouse wheel

```
        long _stdcall PageCount();
```

Return preview pages count

```
        TxfrxPreviewTool _stdcall Tool();
        void _stdcall Tool([in] TxfrxPreviewTool rhs);
```

Controls preview tool

```
        double _stdcall Zoom();
        void _stdcall Zoom([in] double rhs);
```

Controls Zoom value

```
        frxZoomMode _stdcall ZoomMode();
        void _stdcall ZoomMode([in] frxZoomMode rhs);
```

Controls Zoom mode

```
        VARIANT_BOOL _stdcall OutlineVisible();
        void _stdcall OutlineVisible([in] VARIANT_BOOL rhs);
```

```
Controls Outline visibilty

        long _stdcall OutlineWidth();
        void _stdcall OutlineWidth([in] long rhs);

Controls outline width

        VARIANT_BOOL _stdcall DoubleBuffered();
        void _stdcall DoubleBuffered([in] VARIANT_BOOL rhs);

Controls double buffered property

        VARIANT_BOOL _stdcall AlignDisabled();
        long _stdcall VisibleDockClientCount();
        long _stdcall DrawTextBiDiModeFlagsReadingOnly();
        VARIANT_BOOL _stdcall Enabled();
        void _stdcall Enabled([in] VARIANT_BOOL rhs);
        VARIANT_BOOL _stdcall IsRightToLeft();
        VARIANT_BOOL _stdcall UseRightToLeftReading();
        VARIANT_BOOL _stdcall UseRightToLeftScrollBar();
        VARIANT_BOOL _stdcall Visible();
        void _stdcall Visible([in] VARIANT_BOOL rhs);
        void _stdcall SetSubComponent([in] VARIANT_BOOL IsSubComponent);

Some ActiveX options

        IfrxReport* _stdcall Report();
        void _stdcall Report([in] IfrxReport* rhs);

Controls link between report object and preview object.

        void _stdcall LoadPreparedReportFromStream([in] IUnknown* Stream);

Allow to load prepared report from COM or .NET stream.

};
```

TfrxActivePreviewForm

TfrxActivePreviewForm ActiveX control provides the means to display preview form to display prepared reports. It includes additional user interface controls such as zoom, exports, navigation buttons, and other controls. TfrxPreviewX object implements two interfaces IfrxActivePreviewForm and IfrxActivePreviewFormEvents events.

```
dispinterface IfrxActivePreviewForm {

        VARIANT_BOOL AutoScroll();
        void AutoScroll([in] VARIANT_BOOL rhs);
        VARIANT_BOOL AutoSize();
        void AutoSize([in] VARIANT_BOOL rhs);
        TxActiveFormBorderStyle AxBorderStyle();
        void AxBorderStyle([in] TxActiveFormBorderStyle rhs);
        BSTR Caption();
        void Caption([in] BSTR rhs);
        OLE_COLOR Color();
```

```
        void Color([in] OLE_COLOR rhs);
        IFontDisp* Font();
        void Font([in] IFontDisp* rhs);
        void Font([in] IFontDisp** rhs);
        VARIANT_BOOL KeyPreview();
        void KeyPreview([in] VARIANT_BOOL rhs);
        long PixelsPerInch();
        void PixelsPerInch([in] long rhs);
        TxPrintScale PrintScale();
        void PrintScale([in] TxPrintScale rhs);
        VARIANT_BOOL Scaled();
        void Scaled([in] VARIANT_BOOL rhs);
```

Methods and properties above are ActiveX related methods. Refer to ActiveX documentation for detail description.

```
        void LoadPreparedReport([in] BSTR Value);
```

Loads prepared report into ActiveX preview form.

```
        IfrxReport* Report();
```

Returns report object as its interface.

```
        void Report([in] IfrxReport* rhs);
```

Assigns report object to rpeview form by its interface.

```
};
```

TfrxGzipCompressor object

TfrxGzipCompressor provides the means to compress and decompress streams. It supports two types of streams - COM streams and .NET streams. This object implements IfrxCustomCompressor interface with two methods.

```
void _stdcall CompressStream(
    [in] IUnknown* InputStream,
   [in] IUnknown* OutputStream,
   [in] long CompressionRate,
   [in] BSTR FileName);
```

This method compress stream from InputStream into OutputStream. Compression rate is the value which of compreesion rate. Zero means no compression and 3 means maximum compression. FileName argument is the name of file which is stored into archive as original file name.

```
void _stdcall DecompressStream([in] IUnknown* InputStream);
```

This method decompresses input stream to a disk file with name, which set by FileName argument of CompressStream method.

TfrxReportClient object

The TfrxReportClient provides the means to communicate with the FastReport Server. It can establish connection to server, request report processing, and load prepared report into TfrxReport objects. TfrxReportClient supports IfrxReportClient interface with following methods and properties:

```
IfrxReportServerConnection* ReportServerConnection();
```

The property that gets the server connection interface, which describes all connection's properties between client and FR server. See description below.

```
void PrepareReport();
```

Executes report on FR server and downloads it to a client.

```
void ShowReport();
```

Locally show report, which has been prepared on remote server.

```
BSTR ReportName();
```

Return report name

```
void ReportName([in] BSTR rhs);
```

Set report name

```
BSTR GetServerVariable([in] BSTR VariableName);
```

Request variable value from FR server

To establish connection between Report Client and FR server several properties must be assigned. These properties controlled by IfrxReportServerConnection interface that can be obtained by ReportServerConnection() method of IfrxReportClient interface.

```
interface IfrxReportServerConnection : IDispatch {

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

Enables or disables the data compression between Client and FR Server.

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

This property controls the host name of FR Server. Value can be either the domain name or IP address.

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

This property controls the login name for the connection authorization access to the FR Server

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

This property enables or disables MD5 checksum verification of data transmitting between client and server.

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

This property controls the password for the connection authorization access to the FR Server

```
    HRESULT _stdcall Port([out, retval] long* Value);
    HRESULT _stdcall Port([in] long Value);
```

This property controls TCP port number for TCP connection to FR Server

```
    HRESULT _stdcall ProxyPort([out, retval] long* Value);
    HRESULT _stdcall ProxyPort([in] long Value);
```

This property controls the port of proxy server (in case of proxy server is using)

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

This property controls the proxy server host name or (in case of proxy server is using). Empty string value disables using proxy.

```
    HRESULT _stdcall RetryCount([out, retval] long* Value);
    HRESULT _stdcall RetryCount([in] long Value);
```

This property controls of retry count in case of communication problems with FR Server.

```
    HRESULT _stdcall RetryTimeout([out, retval] long* Value);
    HRESULT _stdcall RetryTimeout([in] long Value);
```

This property controls time interval between retrying

```
    HRESULT _stdcall Timeout([out, retval] long* Value);
    HRESULT _stdcall Timeout([in] long Value);
```

This property controls operation timeout.

```
};
```

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(true) == 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.FP3");
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 - it capably to export in number of common used formats, such as PDF, bitmap images, simple text and others.

Note: Starting from build 3.21.50 the PageNumbers property of IfrxPrintOptions interface also affects to all exports.

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.

```
pRepot->PrintOptions->PageNumbers = _bstr_t("");
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.PrintOptions.PageNumbers = "";
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.

```
    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. Refer to table below for arguments description:

Name	Type	Description
FileName	String	The name of file(s) for export
Pictures	Boolean	Enable or disable export pictures
FixedWidth	Boolean	If true then exported pages has a fixed width
Multipage	Boolean	Put report pages into separated HTML files
Navigatur	Boolean	If true then export additional navigator page
PicsInSameFolder	Boolean	If true then place pictures into same folder
Background	Boolean	If true then export report background too

```
    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);
```

Exports prepared report to the MS Excel format. This type export requires the MS Excel installed on computer.

```
    HRESULT 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.

```
    HRESULT 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 JPEG image.

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

Exports prepared report to TIFF image.

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

Exports prepared report to simple ASCII text.

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

Exports prepared report to CSV format.

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

Exports prepared report to the GIF image.

```
    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);
```

Sends E-mail. This gives ability of sending E-mail by SMTP protocol. Refer table below for arguments description:

Name	Type	Description	Example
Server	String	URL or IP address of SMTP server	"mail.fast-report.com"
Port	Integer	TCP port of SMTP server	25
User	String	User name for SMTP authorization	"mailbot"
Password	String	Password for SMTP authorization	"Hjk344Rd"
From	String	Sender E-mail address	"mailbot@fast-report.ru"
Recipient	String	Recipient E-mail address	"boss@fast-report.com"
Subject	String	Mail subject string	"Weekly report"

Text	String	Mail text strings	"Hello Boss!\n\nBye!"
FileName	String	File name on the disk for attach	"C:\\Reports\\Sell.fp3"
AttachName	String	The name of attach for recipient	"NewSells.fp3"

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.

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.**

In order to support Master/Detail relation between .NET tables following algorithm may be useful:

1. Each record of the master table has a unique key for the detail table.
2. Detail data must be a FrxDataView type.
3. Every navigation event between the master table records should set a new filter on the detail table. Application must catch OnFirst, OnNext, and OnPrior events of master FrxDataTable or master FrxDataView object.
4. In the event handler of these event set a filter on the detail DataView object.

Following code demonstrates how to set Master/Detail relation for ADO .NET objects. The datatable is master FrxDataTable object and dataview is the detail FrxDataView object. Both objects related by "id" field.

```
private void datatable_FrxEventHandler()
{
    object id;

    // Find ID value of currnt record of master table
    datatable.OnGetValueHandler("id", out id);
    // Select records of detail data with given ID
    dataview.RowFilter = string.Format("id = {0}", id );
}
```

Event registration code:

```
    datatable = new FrxDataTable("DataTableDemo");
    detail_table = new FrxDataTable("DetailTable");
    FillTableWithSampleData(datatable);

    // These events used for Master/Detail implementation
    datatable.FrxEventOnFirst += new FrxOnFirst(datatable_FrxEventHandler);
    datatable.FrxEventOnNext += new FrxOnNext(datatable_FrxEventHandler);
    datatable.FrxEventOnPrior += new FrxOnPrior(datatable_FrxEventHandler);
```

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";
```

Finally, new method added to IfrxReport interface – FindObject. This method allows finding any report object by its name. For example, to find "Memo3" object just use following form:

```
IfrxMemoView  memo = report.FindObject("Memo3") as IfrxMemoView;
```

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
IfrxSubreport	Creates subreport object
IfrxHeader	Create header band
IfrxFooter	Creates footer band
IfrxMasterData	Creates Master data band
IfrxDetailData	Creates detail data band
IfrxSubdetailData	Creates subdetail data band
IfrxDataBand4	Creates the 4-th level embedded data band
IfrxDataBand5	Creates the 5-th level embedded data band
IfrxDataBand6	Creates the 6-th level embedded data band
IfrxPageHeader	Creates page header
IfrxPageFooter	Creates page footer
IfrxColumnHeader	Creates column header
IfrxColumnFooter	Creates column footer
IfrxGroupHeader	Creates group header
IfrxGroupFooter	Creates group footer
IfrxChild	Creates child band
IfrxOverlay	Creates overlay band

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
```

Visual Basic 6

Creates report object with VB 6 is different. Due to impossibility of using UUID in VB6 the **`CreateReportObjectEx`** must be used instead of **`CreateReportObject`**.

```
Function CreateReportObjectEx(ParentObject As IfrxComponent, ObjectType As
String, Name As String) As IfrxComponent
```

This method is similar to CreateReportObject except that the ObjectType argument is a string, which represent object name. There a simple rule for convert interface name to an ObjectType. All object type name make from interface name by changing leading 'I' character to the 'T' character. For example, IfrxMasterData -> TfrxMasterData, IfrxMemmoView -> TfrxMemoView. Code below illustrates creatation report page.

```
Dim WithEvents report As FastReport.TfrxReport
Dim page1 As FastReport.IfrxreportPage
Set report = CreateObject("FastReport.TfrxReport")
page1 = report.CreateReportObjectEx( report, "TfrxReportPage", "ReportPage1")
```

Objects hierarchy

This topic describes an object's hierarchy of Fast Report Studio interfaces. The means of this topic is to provide the object's inheritance relation information to the application programmer. The FR Studio designed in manner that most interfaces implements only object specific properties and methods. To use properties and methods that are more common you may need request the parent object's interface. Let us give somes examples how to request interface in a different programming languages. The following examples imply that `memo` is an object that defined by IfrxMemoView interface.

C#:

```
(memo as IfrxComponent).Height = 35.3;
```

C++ raw example:

```
IfrxComponent*  pComponent;
memo->QueryInterface(__uuidof(IfrxComponent), (void**) &pComponent);
pComponent->Height = 35.3;
pComponent->Release();
```

C++ using AutoPtr approach:

```
IfrxComponentPtr component = memo;
component->Heght = 35.3;
```

VB.NET:

```
CType(memo, IfrxComponent).Heght = 35.3
```

VB6:

```
Dim component As IfrxComponent
component = memo
memo.Height = 35.3
component = Nothing
```

Finally, see the "Objects hierarchy" outline tree to see object inheritance layout.

IfrxComponent

The IfrxComponent is the base interface for many objects of the Fast Report. It consists of several properties and methods that responsible for object's position in the report. Thereferoe, to set object size and position a programmer should obtain this interface from object and set it properties. In addition, IfrxComponent has a name property, which stores an object user name that assigned to object, f.e. "Memo1", "Memo2", "MasterData1", and so on.
Finally, the createReportObject method of IfrxReport interface getd this interface for any newly

created object. To set object specific properties you need to obtain object specific interface in manner described above.

```
interface IfrxComponent : IDispatch {

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

This property gets object base name

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

This property gets string with object description.

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

This property returns the child objects count.

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

This property returns the left coordinate of an object.

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

This property sets the left coordinate of an object.

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

This property returns the top coordinate of an object.

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

This property sets the top coordinate of an object.

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

This property returns an object's width.

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

This property sets an object's width.

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

This property returns an object's height.

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

This property sets an object's height.

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

This method searches for object by its name. If object found then method returns interface IfrxComponent to that object. Note that this interface шы able to find only child objects of object that calling this method. That means that you cannot find report page if you call this method on band object. We recommend use FindObject method of IfrxReport interface instead of this method.

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

This property gets an alias name of object.

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

This property gets a name of object.

```
};
```

This chapter describes objects that are descendant to IfrxComponent interface.

IfrxReport

This interface aggregates properties and methods of the TfrxReport object – the main one object of FastReport. See "FastReport objects review" part "TfrxReport" chapter for detail description of the TfrxReport and its interfaces. Examples on using TfrxReport could be found in "Working with TfrxReport object" chapter.

IfrxDataSet

IfrxDataSet is the base interface for any data access object of the FastReport. It is not necessary to use this interface to make report, but it can be usefult for access data through biltin FastReport's database engine.

NOTE for .NET programmers: Please keep in mind a differency of terminology between FastReport and ADO.NET. The FR DataSet is analogue of DataTable.NET.

```
interface IfrxDataSet : IDispatch {

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

These properties control a user 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

```
HRESULT _stdcall FieldsCount([out, retval] long* Value);
```

Gets fields count in data set.

```
HRESULT _stdcall RecordsCount([out, retval] long* Value);
```

Gets count of record of dataset.

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

Gets value of field of current record of dataset. Field name is selected by Fieldname argument.

```
HRESULT _stdcall CurrentRecordNo([out, retval] long* Value);
```

Gets current record's number number.

```
HRESULT _stdcall GoFirst();
```

Go to first datasets's record.

```
HRESULT _stdcall GoNext();
```

Go to next datasets's record.

```
HRESULT _stdcall GoPrior();
```

Go to prior dataets's record.

```
};
```

IfrxUserDataSet

This interface aggregates properties of the TfrxUserDataSet object. See "FastReport objects review" part "TfrxUserDataset" chapter for detail description of the TfrxUser dataset and its

interfaces. This interface is descendant of IfrxDataSet interface.

IfrxADOTable

This interface aggregates properties and methods of the TfrxADOTable object. See "FastReport objects review" part "TfrxADOTable" chapter for detail description of the TfrxADOTable dataset and its interfaces. This interface is descendant of IfrxDataSet interface.

IfrxADOQuery

This interface aggregates properties and methods of the TfrxADOQuery object. See "FastReport objects review" part "TfrxADOQuery" chapter for detail description of the TfrxADOQuery dataset and its interfaces. This interface is descendant of IfrxDataSet interface.

IfrxADODatabase

This interface aggregates properties and methods of the TfrxADODatabase object. See "FastReport objects review" part "TfrxADODatabase" chapter for detail description of the TfrxADODatabase object and its interfaces.

IfrxPage

This interface is the base interface for IfrxReportPage. It includes only single property, which controls page visibility.

```
interface IfrxPage : IUnknown {

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

Gets page visibility attribute

```
    HRESULT _stdcall Visible([in] VARIANT_BOOL Value);
```

Sets page visibility attribute

```
};
```

IfrxReportPage

This interface controls the report page layout and properties. This interface controls a page template, not a prepared page. Note that single template page can generate many prepared page because bands can include many data records.

```
interface IfrxReportPage : IDispatch {
```

```
HRESULT _stdcall SetDefaults();
```

Reset page properties to default valaues

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

Contrls printer's tray for priniting first page.

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

Contrls printer's tray for priniting other pages. Other page can appear if DataBand includes lot of records, so template page can generate many prepared pages.

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

Controls bottom margins

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

Controls colmn count on the page.

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

Controls column's width.

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

Controls column's position. The format of this property is a string with CR/LF terminated fields. Each field is string representation of the foat number.

```
HRESULT _stdcall DataSet([out, retval] IfrxDataSet** Value);
HRESULT _stdcall DataSet([in] IfrxDataSet* Value);
```

This property controls the page's dataset.

```
HRESULT _stdcall Duplex([out, retval] frxDuplexMode* Value);
HRESULT _stdcall Duplex([in] frxDuplexMode Value);
```

Controls page`s printing mode. Dumplex mode can be on of following values.

dmNone	No duplex mode
dmVertical	Vertical duplex
dmHorizontal	Hprizontal duplex
dmSimplex	Vertical and horizontal duplex

Note that this propery affects only to printers that support this mode.

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

Controls horizontal guides for designer mode. The format of this property is a string with CR/LF

terminated fields. Each field is string representation of the foat number.

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

This property allows change page scale in design mode for manually fix bands that are exids of the page bounds.

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

Controls value of left margin.

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

Controls margin's mirroring.

```
HRESULT _stdcall Orientation([out, retval] PrinterOrientation* Value);
HRESULT _stdcall Orientation([in] PrinterOrientation Value);
```

Controls page orientation, i.e. portrait or landscape.

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

Gets or sets outline text.

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

Enables/disables printing empty page.

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

This property allows economizing on paper. For example, if two pages have bands that an occupied only half page and this property set to TRUE, then FR will put both bands into single page.

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

Controls right margin

```
HRESULT _stdcall SubReport([out, retval] IfrxSubreport** Value);
HRESULT _stdcall SubReport([in] IfrxSubreport* Value);
```

Controls subreport object that is IfrxView object, which embed IfrxReportPage object. See IfrxSubreport description for additional information.

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

By default band's title printed after band's header. This property reverses order of band's title

and band's header.

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

Controls top margin.

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

Controls vertical guides for designer mode. The format of this property is a string with CR/LF terminated fields. Each field is string representation of the foat number.

```
HRESULT _stdcall BackPickture([out, retval] OLE_HANDLE* Value);
HRESULT _stdcall BackPickture([in] OLE_HANDLE Value);
```

This property controls page background picture. Value type is a picture handle.
};

IfrxView

This is an ancestor interface for many report objects that printning on page or bands. It stores properties that are common for descendant objects.

```
interface IfrxView : IDispatch {

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

Gets/sets the name of the field of view's dataset.

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

Gets/sets an user defined tag. This tag is very useful for interactive reports.
For example, tag field can be filled in FastScript or in OnBeforePrint event whith unique value. This value can be used later in OnClieckObject event. You can use this property for any other purposes.

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

Gets/sets URL property. If this property is set and user click in preview window on an object, which is descendant of IfrxView, then this URL opens in the default explorer window.

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

Gets/sets the view's dataset by its name.

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

Gets the user defined object's name.

```
    HRESULT _stdcall Frame([out, retval] IfrxFrame** Value);
```

The read-only property, which returns interface to the frame attributes of the object. See description of IfrxFrame interface in 'Auxilary interfaces' chapter.

```
    HRESULT _stdcall ShiftMode([out, retval] frxShiftMode* Value);
    HRESULT _stdcall ShiftMode([in] frxShiftMode Value);
```

This property controls the shift behavior of the object.

ms_DontShift	
sm_ShiftAlways	
sm_WhenOverlapped	

```
    HRESULT _stdcall Align([out, retval] frxAlign* Value);
    HRESULT _stdcall Align([in] frxAlign Value);
```

This property determines the alignment of the object relative to band or page.

ba_None	
ba_Left	
ba_Right	
ba_Center	
ba_Width	
ba_Bottom	
ba_Client	

```
};
```

IfrxMemoView

This interface describes the memo-view objects. The purpose of MemoView is displaying text, database field values, expressions, and so on. This interface is descendant of IfrxView interface. IfrxMemoView interface can be used as argument of CreateObject method of IfrxReport interface, so MemoView object can be created dynamically by application.

Note that the memo-views that are located on the data bands are drawing as many times as count of records in the dataset assigned to that band.

```
interface IfrxMemoView : IDispatch {

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

Gets/sets the automatical width attribute. If this property set to true, then memo's width changed to fit width of its text.

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

This property determines whether the text object may contain expressions inside the text. Default value is true. If this properties is set to false, then no any expressions wil be calculated in this field – it disables using variables, FastScript expressions, and other data transformation. I.e. this

memo will be shown as is.

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

This property determines whether the text object may contain expressions inside the text.

```
HRESULT _stdcall BrushStyle([out, retval] TfrxBrushStyle* Value);
HRESULT _stdcall BrushStyle([in] TfrxBrushStyle Value);
```

The style of the brush used for object's background. Avaulable brush styles described in table below.

SolidBrush	
ClearBrush	
GorizontalBrush	
VericalBrush	
FDiagonalBrush	
BDiagonalBrush	
CrossBrush	
DiagCrossBrush	

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

This properties control amount of pixels between two characters.

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

Determines whether the text should be clipped inside the objects bounds

```
HRESULT _stdcall Color([out, retval] long* Value);
HRESULT _stdcall Color([in] long Value);
```

Get/set color value of the memo's text. Color is 32-bit value. See "MSDN Platform SDK" document, "RGB" topic for description of this value.

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

This property specifies the field from which the object gets data.

```
HRESULT _stdcall DataSet([out, retval] IfrxDataSet** Value);
HRESULT _stdcall DataSet([in] IfrxDataSet* Value);
```

Gets/sets memo's dataset. Description of IfrxDataSet interface given in "Objects hierarchy" chapter.

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

Gets memo's dataset name.

```
HRESULT _stdcall DataSetName([in] BSTR Value);
```

Set memo's dataset by its name.

```
HRESULT DisplayFormat([out, retval] IfrxDisplayFormat** Value);

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

This property controls expression delimiters in memo's expression.

```
HRESULT _stdcall FlowTo([out, retval] IfrxMemoView** Value);
HRESULT _stdcall FlowTo([in] IfrxMemoView* Value);
```

This property controls the text object that will show the text that not fit in the current object.

```
HRESULT _stdcall Font([out, retval] IfrxFont** Value);
```

The read-only property, which returns interface to the font attributes of the object. See description of IfrxFont interface in 'Auxilary interfaces' chapter.

```
HRESULT _stdcall Frame([out, retval] IfrxFrame** Value);
```

The read-only property, which returns interface to the frame attributes of the object. See description of IfrxFrame interface in 'Auxilary interfaces' chapter.

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

This property controls the left indent of the text inside memo.

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

This property controls the top indent of the text inside memo.

```
HRESULT _stdcall HAlign([out, retval] frxHAlign* Value);
HRESULT _stdcall HAlign([in] frxHAlign Value);
```

Controls horizontal align of the memo's text. Align modes are described in the table below.

hAlignLeft	Horizontal align memo's text to left
hAlignRight	Horizontal align memo's text to right
hAlignCenter	Horizontal align memo's text in center
hAlignBlock	Justify text in memo

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

This property controls displaying zeros on the memo. This property is meaningless for non-numeric memos formats.

```
HRESULT _stdcall Highlight([out, retval] IfrxHighlight** Value);
```

The read-only property, which returns interface to the conditional highlight attributes. See description of IfrxHighlight interface in 'Auxilary interfaces' chapter.

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

Gets/sets interval between lines.

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

This property holds memo text. Value could be any text or expression.

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

This property controls gap between paragraphs.

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

This property controls using parent object's font for memo. If propery is set to true, then memom using parent object's font.

```
HRESULT _stdcall Rotation([out, retval] long* Value);
HRESULT _stdcall Rotation([in] long Value);
```

This attribute controls rotation angle. The value unit is degree.

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

This attribute controls right-to-left reading mode for memo.

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

This property controls memo style.

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

This attribute controls suppresses repeated values.

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

Control memo's text underline attribute.

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

This attribute controls the break Russian words.

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

This attribute determines whether the text object inserts soft carriage returns so text wraps at the right margin.

```
HRESULT _stdcall VAlign([out, retval] frxVAlign* Value);
HRESULT _stdcall VAlign([in] frxVAlign Value);
```

Gets/sets vertical align mode. Align modes are described in the table below.

vAlignTop	Vertical align memo's text to the top.
vAlignBottom	Vertical align memo's text to the bottom.
vAlignCenter	Vertical align memo's text to the center.

```
frxStretchMode _stdcall StretchMode();
void _stdcall StretchMode([in] frxStretchMode rhs);
```

Gets/sets view's stretch mode. Stretch mode described in the table below.

sm_DontStretch	Do not stretch view – use actual size of view
sm_ActualHeight	Change view's height to show all contents of the view.
sm_MaxHeight	Maximize view's height until top bound of the next object.

};

IfrxPictureView

This interface controls PictureView objects of the report. These objects used for displaying pictures on the report. This interface is descendant of IfrxView interface. This interface can be used as argument in CreateObject method of IfrxReport interface, so PictureView object can be created dynamically by application.

```
interface IfrxPictureView : IDispatch {

    HRESULT _stdcall Picture([out, retval] OLE_HANDLE* Value);
    HRESULT _stdcall Picture([in] OLE_HANDLE Value);
```

This property provides access to picture by its handle. You can set pictures by means of aplication by setting handle value in OnBeforePrintEvent of IfrxPictureView object.

```
    HRESULT _stdcall LoadViewFromStream([in] IUnknown* Stream);
    HRESULT _stdcall SaveViewToStream([in] IUnknown* Stream);
```

This interface provides access to picture over streams. You can set pictures by means of application by calling LoadViewFromStream method in OnBeforePrintEvent of IfrxPictureView object. This method supports COM and .NET streams.

};

IfrxChartView

This interface controls the built-in charts object, which implemented on top of TeeChart library. This interface is descendant of IfrxView interface. IfrxChartView interface can be used as argument of CreateObject method of IfrxReport interface, so ChartView object can be created dynamically by application.

Note that ChartView interfaces provide limited base functionality. Most of properties you can set only in FR Designer mode.

```
interface IfrxChartView : IDispatch {
```

```
HRESULT _stdcall GetSeriesItem(
                [in] long Index,
                [out, retval] IfrxSeriesItem** Value);
```

Get chart series item by its number. The out value is the IfrxSeriesItem interface to the series object. See description of IfrxSeriesItem interface below.

```
HRESULT _stdcall AddSeriesItem(
                [in] frxSeriesType SeriesType,
                [out, retval] IfrxSeriesItem** NewItem);
```

Add new series item into ChartView and return its interface. Series item type is set by SeriesType argument, which have one of following values:

frxSeriesLine, frxSeriesArea, frxSeriesPoint, frxSeriesBar, frxSeriesHorizBar, frxSeriesPie, frxSeriesGantt, frxSeriesFastLine, frxSeriesArrow, frxSeriesBubble, frxSeriesChartShape, frxSeriesHorizArea, frxSeriesHorizLine, frxSeriesPolar, frxSeriesRadar, frxSeriesPolarBar, frxSeriesGauge, frxSeriesSmith, frxSeriesPyramid, frxSeriesDonut, frxSeriesBezier, frxSeriesCandle, frxSeriesVolume, frxSeriesPointFigure, frxSeriesHistogram, frxSeriesHorizHistogram, frxSeriesErrorBar, frxSeriesError, frxSeriesHighLow, frxSeriesFunnel, frxSeriesBox, frxSeriesHorizBox, frxSeriesSurface, frxSeriesContour, frxSeriesWaterFall, frxSeriesColorGrid, frxSeriesVector3D, frxSeriesTower, frxSeriesTriSurface, frxSeriesPoint3D, frxSeriesBubble3D, frxSeriesMyPoint, frxSeriesBarJoin, frxSeriesBar3D,

The out value is the IfrxSeriesItem interface to the series object.

```
HRESULT _stdcall SeriesCount([out, retval] long* Value);
```

This read-only property gets count of series item into ChartView object.

```
HRESULT _stdcall View3D([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall View3D([in] VARIANT_BOOL Value);
```

This property controls the 3D depth visual effect of the chart.

```
HRESULT _stdcall View3dWalls([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall View3dWalls([in] VARIANT_BOOL Value);
```

This property controls the 3D depth visual effect for the chart walls.

```
HRESULT _stdcall LeftAxis([out, retval] IfrxChartAxis** Value);
HRESULT _stdcall BottomAxis([out, retval] IfrxChartAxis** Value);
```

These two properties return interfaces for Lft and bottom axis respectively.
};

The chart series defined by IfrxSeriesItem interface and controlled through it. The methods and properties provided by IfrxSeriesItem are common for all series types.

```
interface IfrxSeriesItem : IUnknown {

    HRESULT _stdcall DataBand([out, retval] IfrxDataBand** Value);
    HRESULT _stdcall DataBand([in] IfrxDataBand* Value);

    HRESULT _stdcall DataSet([out, retval] IfrxDataSet** Value);
    HRESULT _stdcall DataSet([in] IfrxDataSet* Value);
```

This property controls charts's dataset. Description of IfrxDataSet interface given in "Objects hierarchy" chapter.

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

Gets chart's dataset name.

```
HRESULT _stdcall DataSetName([in] BSTR Value);
```

Set chart's dataset by its name.

```
HRESULT _stdcall XSource([out, retval] BSTR* Value);
HRESULT _stdcall XSource([in] BSTR Value);
HRESULT _stdcall YSource([out, retval] BSTR* Value);
HRESULT _stdcall YSource([in] BSTR Value);
HRESULT _stdcall ZSource([out, retval] BSTR* Value);
HRESULT _stdcall ZSource([in] BSTR Value);
HRESULT _stdcall FourthSource([out, retval] BSTR* Value);
HRESULT _stdcall FourthSource([in] BSTR Value);
HRESULT _stdcall FifthSource([out, retval] BSTR* Value);
HRESULT _stdcall FifthSource([in] BSTR Value);
HRESULT _stdcall SixthSource([out, retval] BSTR* Value);
HRESULT _stdcall SixthSource([in] BSTR Value);
```

These properties controls data source for sieries item for X, Y, Z, and other dimensions respectively.

```
HRESULT _stdcall XValues([out, retval] BSTR* Value);
HRESULT _stdcall XValues([in] BSTR Value);
HRESULT _stdcall YValues([out, retval] BSTR* Value);
HRESULT _stdcall YValues([in] BSTR Value);
HRESULT _stdcall ZValues([out, retval] BSTR* Value);
HRESULT _stdcall ZValues([in] BSTR Value);
HRESULT _stdcall FourthValues([out, retval] BSTR* Value);
HRESULT _stdcall FourthValues([in] BSTR Value);
HRESULT _stdcall FifthValues([out, retval] BSTR* Value);
HRESULT _stdcall FifthValues([in] BSTR Value);
HRESULT _stdcall SixthValues([out, retval] BSTR* Value);
HRESULT _stdcall SixthValues([in] BSTR Value);
```

These properties hold the values for series item for X, Y, Z, and other dimensions respectively. Note that these functions are alternative for "Source" properties and can be used for manually set values to series.

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

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

This property controls series item title. It will appear on legend window.

```
};

interface IfrxChartAxis : IDispatch {
```

This interface controls the chart axis.
Note that documentation updating is much rare of FR Studio source code. So at time you are riding this manual new properties of methods cam be added to this interface.

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

This property controls the automatic zoom of axis. The default value is true. Set value to false before setting minimum and maximum values. Minimum and maximum values are usefule for manual chart zooming.

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

This property controls the minimum value of the axis.

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

This property controls the maxim value of the axis.

```
};
```

IfrxShapeView

This interface controls the ShapeView object. This interface is descendant of IfrxView interface. IfrxShapeView interface can be used as argument of CreateObject method of IfrxReport interface, so ShapeView object can be created dynamically by application.

```
interface IfrxShapeView : IDispatch {

    HRESULT Curve([out, retval] long* Value);
    HRESULT Curve([in] long Value);

    HRESULT _stdcall ShapeType([out, retval] frxShapeType* Value);
    HRESULT _stdcall ShapeType([in] frxShapeType Value);
```

This property controls shape type. The ShapeType can take the one of the folowing values: *skRectangle, skRoundRectangle, skEllipse, skTriangle, skDiamond, skDiagonal1, skDiagonal2*.

```
    HRESULT _stdcall Frame([out, retval] IfrxFrame** Value);
```

The read-only property, which returns interface to the frame attributes of the object. See description of IfrxFrame interface in 'Auxilary interfaces' chapter.

```
};
```

IfrxOLEView

This interface controls the OLEView object. This interface is descendant of IfrxView interface. IfrxOLEView interface can be used as argument of CreateObject method of IfrxReport interface,

so ShapeView object can be created dynamically by application.

```
interface IfrxOLEView : IDispatch {

    HRESULT _stdcall OleContainer([out, retval] IUnknown** Value);
```

This read-only propery gives interface to the OLE container which is standard ActiveX interface.

```
    HRESULT SizeMode([out, retval] long* Value);
    HRESULT SizeMode([in] long Value);

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

This property determines whether the object can be stretched

```
};
```

IfrxRichView

This interface controls the RichView object. This interface is descendant of IfrxView interface. IfrxRichView interface can be used as argument of CreateObject method of IfrxReport interface, so ShapeView object can be created dynamically by application.

```
interface IfrxRichView : IDispatch {

    HRESULT _stdcall LoadViewFromStream([in] IUnknown* Stream);
```

Load RichTtextFormat (RTF) stream into RichView object.

```
    HRESULT _stdcall SaveViewToStream([in] IUnknown* Stream);
```

Save RichTtextFormat (RTF) from RichView object to stream.

```
};
```

IfrxSubreport

This interface controls the Subreport object. Subreport object is a report page that embedded into Subreport object. This interface is descendant of IfrxView interface. IfrxSubreport interface can be used as argument of CreateObject method of IfrxReport interface, so Subreport object can be created dynamically by application.

```
interface IfrxSubreport : IDispatch {

    HRESULT _stdcall Page([out, retval] IfrxReportPage** Value);
    HRESULT _stdcall Page([in] IfrxReportPage* Value);
```

This property controls the subreport page object which given as IfrxReportPage interface. IfrxReportPage interface described in "Object hierarchy" chapter.

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

This property controls The PrintOnParent attribute.

```
};
```

IfrxBand

IfrxBand interface is an ancestor for all band objects. It stores properties that are common for descendant objects.

```
interface IfrxBand : IDispatch {

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

This property determines whether the band may split its contents across pages

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

This property determines whether the band will be printed together with its child

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

This property controls the text that will be shown in the preview outline control

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

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

If this property is set, then it starts a new page before printing a band.

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

This property determines whether the object can be stretched

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

This property determines whether the child band will be printed if its parent band is invisible.

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

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

IfrxDataBand

This interface controls the DataBand object.

```
interface IfrxDataBand : IDispatch {
```

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

This property controls the gap between band columns.

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

This property controls the width of the band column

```
HRESULT _stdcall ColumnsCount([out, retval] long* Value);
HRESULT _stdcall ColumnsCount([in] long Value);
```

This property controls number of columns in the band

```
HRESULT _stdcall CurrentColumn([out, retval] long* Value);
HRESULT _stdcall CurrentColumn([in] long Value);

HRESULT _stdcall DataSet([out, retval] IfrxDataSet** Value);
HRESULT _stdcall DataSet([in] IfrxDataSet* Value);
```

This property controls the page's dataset.

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

This property determines whether the footer band should be shown after each data row

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

This property determines whether the band will be printed together with its footer

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

This property determines whether the band will be printed together with its header

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

This property determines whether the band will be printed together with all its subbands

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

This property determines whether the databand will be printed if its subband is empty

```
HRESULT _stdcall RowCount([out, retval] long* Value);
HRESULT _stdcall RowCount([in] long Value);
```

This property controls number of virtual records in the databand

```
HRESULT _stdcall ResetDataSet();
```

In opposite to DataSet property this method unlink dataset from the band.

```
};
```

IfrxMasterData

The master data band object.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxMasterData is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxDetailData

The detail data band object.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxDetailData is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxSubdetailData

The subdetail data band object.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxSubdetail is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxDataBand4

The 4-th level embedded data band.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxDataBand4 is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxDataBand5

The 5-th level embedded data band.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxDataBand5 is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxDataBand6

The 6-th level embedded data band.
This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxDataband6 is a descendant of IfrxDataBand, so IfrxDataBand interface can be requested and all its properties will be available.

IfrxReportTitle

This interface does not provide any methods or or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report title by application code. IfrxReportTitle is a descendant of IfrxBand, so IfrxBand interface can be requested and all its properties will be available.

IfrxReportSummary

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxReportSummary is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

IfrxHeader

```
interface IfrxHeader : IDispatch {
    HRESULT _stdcall ReprintOnNewPage([out, retval] VARIANT_BOOL* Value);
    HRESULT _stdcall ReprintOnNewPage([in] VARIANT_BOOL Value);
```

This property determines whether the header band will be reprinted on new page

```
};
```

IfrxFooter

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxFooter is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

IfrxPageHeader

```
interface IfrxPageHeader : IDispatch {

    HRESULT PrintOnFirstPage([out, retval] VARIANT_BOOL* Value);
    HRESULT PrintOnFirstPage([in] VARIANT_BOOL Value);
};
```

IfrxPageFooter

```
interface IfrxPageFooter : IDispatch {

    HRESULT PrintOnFirstPage([out, retval] VARIANT_BOOL* Value);
    HRESULT PrintOnFirstPage([in] VARIANT_BOOL Value);

    HRESULT PrintOnLastPage([out, retval] VARIANT_BOOL* Value);
    HRESULT PrintOnLastPage([in] VARIANT_BOOL Value);
};
```

IfrxColumnHeader

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxColumnHeader is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

IfrxColumnFooter

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxColumnFooter is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

IfrxGroupHeader

```
interface IfrxGroupHeader : IDispatch {

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

    HRESULT KeepTogether([out, retval] VARIANT_BOOL* Value);
    HRESULT KeepTogether([in] VARIANT_BOOL Value);

    HRESULT ReprintOnNewPage([out, retval] VARIANT_BOOL* Value);
    HRESULT ReprintOnNewPage([in] VARIANT_BOOL Value);

    HRESULT LastValue([out, retval] VARIANT* Value);
};
```

IfrxGroupFooter

```
interface IfrxGroupFooter : IDispatch {

    HRESULT HideIfSingledatarecord([out, retval] VARIANT_BOOL* Value);
    HRESULT HideIfSingledatarecord([in] VARIANT_BOOL Value);
};
```

IfrxChild

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxChild is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

IfrxOverlay

This interface does not provide any methods or properties itself and used only for CreateObject method of IfrxReport interface. In other words, this interface is mean for creation the report summary by application code. IfrxOverlay is a descendant of IfrxBand, so IfrxBand interface can be requested all its properties will be available.

Auxiliary interfaces

This topic describes auxiliary interfaces that introduce to implement some extended properties of report objects.

IfrxFont

This interface controls the font typeface.

```
interface IfrxFont : IUnknown {

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

This property determines whether the font is the bold.

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

This property controls the font size

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

This property controls the name of font.

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

This property determines whether the font is italic

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

This property determines whether the font is underline.

```
    HRESULT _stdcall Handle([out, retval] long* Value);
    HRESULT _stdcall Handle([in] long Value);
```

This property controls the font handle.

```
};
```

IfrxFrame

This interface controls object frame options.

```
interface IfrxFrame : IUnknown {

    HRESULT _stdcall Color([out, retval] long* Value);
    HRESULT _stdcall Color([in] long Value);
```

This property controls the frame color.

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

This property determines whether the object is drop shadow.

```
    HRESULT _stdcall ShadowColor([out, retval] long* Value);
    HRESULT _stdcall ShadowColor([in] long Value);
```

This property controls color of object's shadow

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

This properties control the width of shadow

```
    HRESULT _stdcall Style([out, retval] long* Value);
    HRESULT _stdcall Style([in] long Value);
```

This property controls the frame's style.

```
    HRESULT _stdcall FrameType([out, retval] long* Value);
    HRESULT _stdcall FrameType([in] long Value);
```

This property controls the frame type

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

This properties control the width of frame.

```
};
```

IfrxHighlight

This interface controls object's highlight options.

```
interface IfrxHighlight : IUnknown {

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

This property determines whether the highlite is active.

```
    HRESULT _stdcall Color([out, retval] long* Value);
    HRESULT _stdcall Color([in] long Value);
```

This property controls the highlite color.

```
    HRESULT _stdcall Font([out, retval] IfrxFont** Value);
```

This read-only property gets IfrxFont interface, which controls highlite font.

```
};
```

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, sometime we are change interfaces (mostly append new methods and properties). As result, you application can works in improper way or raise an 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.