FastReport Studio

Руководство программиста

Редакция 1.10

История исправления документа

1.00	26 Августа 2005	Первоначальный релиз
1.01	26 Сентября 2005	Корректировка раздела экспорта
1.02	26 Сентября 2005	Рассмотрение возможности "Default connection"
1.03	29 Сентября 2005	Рассмотрение аспектов работы в многопоточной среде
1.04	30 Октября 2005	Незначительные изменения
1.05	30 Октября2005	Незначительные изменения
1.06	3 Ноября 2005	Незначительные изменения
1.07	21 Марта 2006	Добавлена иерархия объектов
1.08	31 Марта 2006	Более подробное описание возможностей FR Studio
1.09	5 Июня 2006	Обновление «обёртки» ADO NET Добавлено описание TfrxGzipCompressor Добавлено описание методов IfrxBuiltinExports-> SendMail
1.10	9 Июня 2006	Добавлен раздел "Вспомогательные интерфейсы"
1.11	15 Июня 2006	Добавлено описание всех бэндов
1.12	20 Июня 2006	Добавлено описание IfrxCrossView и IfrxDBCrossView

Table of Contents

Предисловие

Этот документ разбит на нескольких взаимосвязанных разделов, которые описывают различные аспекты использования FastReport Studio.

Раздел «*Обзор объектов FastReport*» описывает COM Co-Classes. Эти классы могут быть созданы непосредственно посредством прикладной программы без использования других классов и интерфейсов. Помимо этого, раздел описывает некоторые интерфейсы, которые не попали в раздел «Иерархия объектов».

Раздел «*Использование объекта TfrxReport*» - аналог Краткого Справочника. Этот раздел содержит примеры, советы и рекомендации по использованию базовых возможностей объекта TfrxReport. Раздел включает примеры на C++, C# и VB.NET. Большая часть примеров приведена для «Managed» и «Unmanaged» сред.

Раздел «*Иерархия объектов*» описывает интерфейсы и информацию об их наследовании. Этот раздел особенно необходим для тех программистов, кто не имеет опыта работы с оригинальным FastReport для Delphi. Основную часть раздела составляет описание объектов, наследованных от IfrxComponent. Также раздел содержит описание вспомогательных интерфейсов, которые управляют различными свойствами объектов.

Обзор объектов FastReport

Продукт FastReport Studio содержит множество объектов, предназначенных для создания отчётов, их модификации, экспорта в различные форматы и расширения функциональности отчётов. В этом разделе описаны некоторые из объектов, доступных в FastReport Studio.

Главный объект генератора отчётов - TfrxReport

Самый главный объект генератора отчётов. Объект TfrxReport предоставляет все необходимые методы и свойства для загрузки, сохранения, дизайна и просмотра отчётов. Каждый TfrxReport объект содержит один отчёт. Объект создаётся непосредственно из прикладной программы, средствами, предоставляемыми средой программирования. Смотрите раздел «Использование объекта TfrxReport», где показаны примеры создания объекта для различных сред программирования.

Наиболее важные методы объекта TfrxReport

```
HRESULT _stdcall ClearReport();
```

Очищает отчёт и его шаблон.

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

Загружает шаблон отчёта из файла, чьё имя заданно параметром szFileName.

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

Загружает шаблон отчёта из потока. Поддерживаются два вида потоков – COM Stream и .NET Stream.

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

Сохраняет шаблон отчёта в файл, чьё имя задано параметром szFileName.

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

Сохраняет шаблон отчёта в поток. Поддерживают два типа потоков – COM Stream и .NET Stream.

```
HRESULT _stdcall DesignReport();
```

Запускает встроенный визуальный дизайнер отчётов.
Внимание: Данные метод не совместим со средой ASP.NET.

```
HRESULT _stdcall ShowReport();
```

Выполняет построение отчёта и отображает построенный отчёт в окне предварительного просмотра.
Внимание: Данные метод не совместим со средой ASP.NET.

```
HRESULT _stdcall PrepareReport([in] VARIANT_BOOL ClearLastReport);
```

Выполняет построение отчёта без его показа в окне предварительного просмотра. Параметр "ClearLastReport" указывает на то, нужно ли чистить ранее подготовленный отчёт. Если он установлен в значение «истина», то ранее подготовленный отчёт очищается, иначе нет. Использование этого параметра позволяет строить композитные отчёты.

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

Отображает заранее подготовленный отчёт в окне предварительного просмотра. Отчёт может быть подготовлен при помощи метода **PrepareReport** или в предыдущем вызове **ShowReport**.

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

Загружает заранее подготовленный отчёт из файла. Впоследствии этот отчёт может быть показан в окне предварительного просмотра или экспортирован в поддерживаемые форматы.

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

Сохраняет подготовленный отчёт в файл, чьё имя задано параметром **szFileName**.

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

Печатает отчёт на принтер. Параметры печати, такие как имя принтера, количество копий и другие, могут быть заданы посредством свойства **PrintOptions** объекта **IfrxReport**.

Способы экспорта отчётов в различные форматы документов

В процессе разработки интерфейсов генератора отчётов, было принято решение вынести экспорты в дополнительный интерфейс. Это решение рассматривалось как временное, но оно прижилось. Итак, все поддерживаемые экспорты вынесены в интерфейс **IfrxBuiltinExports**, который следует запросить у объекта **TfrxReport**. Далее следует описание этого интерфейса. Для примеров экспорта обращайтесь к разделу «Использование объекта TfrxReport».

```
interface IfrxBuiltinExports : IUnknown
{

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

Экспортирует сформированный отчёт в документ формата PDF. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Compressed** указывает, нужно ли сжимать результирующий PDF файл. Аргумент **EmbeddedFonts** указывает, нужно ли внедрять шрифты в документ. Установка этого параметра в значение «истина» значительно увеличивает размер документа, но позволяет его правильно показывать на компьютерах, на которых не установлены соответствующие шрифты.

Аргумент **PrintOptimized** позволяет максимально приблизить вид документа к виду в окне предварительно просмотра, за счёт увеличения размера PDF файла. Оптимальные значения для этих аргументов зависят от различных параметров сформированного отчёта, его сложности и требований к качеству сформированного документа.

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

Экспортирует сформированный отчёт в документ формата XML. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Styles** указывает, нужно ли экспортировать стили. Аргумент **PageBreaks** указывает, нужно ли экспортировать разрывы страниц. Аргумент **WYSIWYG** позволяет максимально приблизить вид экспортируемого документа к виду в окне предварительного просмотра, за счёт увеличения размера экспортируемого файла. Аргумент **Background** указывает, нужно ли экспортировать фоновую картинку. Если отчёт не имеет фоновой картинки, то этот параметр игнорируется.

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

Экспортирует сформированный отчёт в документ формата RTF. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Pictures** указывает, нужно ли экспортировать картинки, присутствующие в отчёте. Если отчёт не имеет картинок, то параметр игнорируется. Аргумент **PageBreaks** указывает, нужно ли экспортировать разрывы страниц. Аргумент **WYSIWYG** позволяет максимально приблизить вид экспортируемого документа к виду в окне предварительного просмотра, за счёт увеличения размера экспортируемого файла.

```
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);
```

Экспортирует сформированный отчёт в документ(ы) формата HTML. Вид и количество экспортируемых страниц зависит от параметров экспорта. Аргумент **FileName** задаёт имя файла(ов), в который(е) происходит экспорт. Аргумент **Pictures** указывает, нужно ли экспортировать картинки, присутствующие в отчёте. Аргумент **FixedWidth** управляет шириной экспортируемого отчёта. При установке этого параметра в значение «истина», используется расширение формата HTML, позволяющее задавать ширину страницы. Аргумент **Multipage** указывает, нужно ли разбивать результирующий документ на страницы. При установке этого параметра в значение «ложь», формируется непрерывная HTML страница отчёта, содержащая весь отчёт. При установке параметра **Multipage** в значение «истина», каждая страница отчёта экспортируется в отдельный HTML файл. Аргумент **Navigator** управляет экспортом навигатора – дополнительного HTML документа, использующего Java

script, который позволяет перемещаться между экспортированными страницами в окне. Аргумент **PicsInSameFolder** управляет размещением экспортируемых картинок. При установке параметра в значение «истина», картинки и результирующие документы экспортируются в одну папку. Аргумент **Background** указывает, нужно ли экспортировать фоновую картинку. Если отчёт не имеет фоновой картинки, то этот параметр игнорируется.

```
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);
```

Экспортирует сформированный отчёт в документ формата Microsoft Excel. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Pictures** указывает, нужно ли экспортировать картинки, присутствующие в отчёте. Аргумент **PageBreaks** указывает, нужно ли экспортировать разрывы страниц. Аргумент **WYSIWYG** позволяет максимально приблизить вид экспортируемого документа к виду в окне предварительного просмотра, за счёт увеличения размера экспортируемого файла. Аргумент **AsText** управляет форматом данных в экспортируемых ячейках. При установке этого параметра в значение «истина» все значения ячеек экспортируются в виде текста. Иначе, FastRepot пытается экспортировать значение в соответствующем формате, например, в числовом. Аргумент **Background** указывает, нужно ли экспортировать фоновую картинку. Если отчёт не имеет фоновой картинки, то этот параметр игнорируется.

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

Экспортирует сформированный отчёт в виде картинки формата BMP. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Monochrome** указывает на необходимость генерации монохромной картинки (чёрно-белой), в противном случае сохраняются оригинальные цвета отчёта. Аргумент **CropPages** позволяет урезать размер картинки до видимой области. Это позволяет уменьшить размер экспортированной картинки. Аргумент **SeparatePages** управляет разбиением на страницы. При установке этого параметра в значение «истина», каждая страница отчёта экспортируется в отдельный графический файл, иначе все страницы экспортируются в один графический файл (режим рулона).

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

Экспортирует сформированный отчёт в виде картинки формата JPEG. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Resolution** управляет качеством результирующей картинки. Значение этого параметра указывает количество точек на дюйм в результирующем графическом файле.

Аргумент **JpegQuality** позволяет устанавливать соотношение между качеством результирующей картинки и размером графического файла. Аргумент **Monochrome** указывает на необходимость генерации монохромной картинки (чёрно-белой), в противном случае сохраняются оригинальные цвета отчёта. Аргумент **CropPages** позволяет урезать размер картинки до видимой области. Это позволяет уменьшить размер экспортированной картинки. Аргумент **SeparatePages** управляет разбиением на страницы. При установке этого параметра в значение «истина», каждая страница отчёта экспортируется в отдельный графический файл, иначе все страницы экспортируются в один графический файл (режим рулона).

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

Экспортирует сформированный отчёт в виде картинки формата TIFF. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент **Resolution** управляет качеством результирующей картинки. Значение этого параметра указывает количество точек на дюйм в результирующем графическом файле. Аргумент **Monochrome** указывает на необходимость генерации монохромной картинки (чёрно-белой), в противном случае сохраняются оригинальные цвета отчёта. Аргумент **CropPages** позволяет урезать размер картинки до видимой области. Это позволяет уменьшить размер экспортированной картинки. Аргумент **SeparatePages** управляет разбиением на страницы. При установке этого параметра в значение «истина», каждая страница отчёта экспортируется в отдельный графический файл, иначе все страницы экспортируются в один графический файл (режим рулона).

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

Экспортирует сформированный отчёт в виде простого ASCII текста. Этот формат не поддерживает экспорт графических изображений и объектов, основанных на графике. Аргумент **FileName** задаёт имя файла, в который происходит экспорт.

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

Экспортирует сформированный отчёт в файл формата CSV. Этот формат не поддерживает экспорт графических изображений и объектов, основанных на графике. Аргумент **FileName** задаёт имя файла, в который происходит экспорт. Аргумент указывает на строку (обычно состоящую из одного символа), который будет использоваться как разделитель. Аргумент **OEMCodepage** указывает на необходимость использования кодовой страницы OEM, вместо текущей кодовой страницы.

};

Доступ к переменным отчёта

Объект TfrxReport предоставляет несколько методов для доступа к переменным отчёта. Переменные отчёта могут быть использованы во встроенном интерпретаторе FastScript, а также могут быть непосредственно выведены в полях Memo.

Следует заметить, что корректная работа с переменными подразумевает первоначальное создание переменной методом AddVariable и последующим изменением переменной методом SetVariable. Нарушение этой последовательности иногда может приводить к нежелательным результатам.

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

Устанавливает значение переменной. Аргумент **Index** указывает имя переменной, Аргумент **Value** указывает на новое значение переменной.

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

Возвращает значение переменной. Аргумент **Index** указывает имя переменной, Аргумент **Value** принимает значение переменной.

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

Добавляет новую переменную в категорию. Если категория не существует, то метод добавляет также и новую категорию. Аргумент **Category** указывает имя категории. Аргумент **Index** указывает имя переменной. Аргумент **Value** принимает значение переменной.

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

Удаляет категорию переменных

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

Удаляет переменную.

Другие полезные методы объекта TfrxReport:

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

Этот метод позволяет привязывать и отвязывать датасеты от шаблона отчёта. Для использования датасета в отчёте он обязательно должен быть привязан к отчёту. Также, привязка датасета к шаблону отчёта позволяет увидеть этот датасет в окне «Поля БД» (окно справа) в режиме дизайнера. Аналогичную операцию можно выполнить в режиме runtime дизайнера посредством меню «Отчёт» -> «Данные» -> «Выбор датасета». Аргумент **Selected** управляет привязкой датасета, значение «истина» привязывает датасет к отчёту, значение «ложь» - отвязывает. Аргумент **DataSet** указывает на любой интерфейс датасета. В качестве датасета могут быть объекты, поддерживающие интерфейс IfrxDataSet: IfrxADOTable, IfrxADOQuery, IfrxUserDataSet.

```
HRESULT _stdcall CreateReportObject(
      [in] IfrxComponent* ParentObject,
```

```
        [in] GUID ObjectType,
        [in] BSTR Name,
        [out, retval] IfrxComponent** GeneratedObject);
```

Этот метод используется для динамического построения шаблона отчёта. Он предоставляет прикладной программе возможность создавать элементы отчёта – страница шаблона отчёта, бэнды, текстовые поля и другие. Аргумент **ParentObject** указывает на родительский объект создаваемого объекта. Аргумент **ObjectType** указывает тип создаваемого объекта. Аргумент **Name** задаёт уникальное имя создаваемого объекта. Аргумент **GeneratedObject** принимает указатель на интерфейс **IfrxComponent** вновь созданного объекта. Список объектов, а также пример использования метода смотрите в разделе «Работа с TfrxReport» глава «Создание шаблона отчёта из кода».

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

Этот метод аналогичен методу **CreateReportObject** за исключением того, что тип создаваемого объекта указывается в виде строкового значения. Этот метод введён для совместимости с Visual Basic 6. Список объектов, а также пример использования метода смотрите в разделе «Работа с TfrxReport» глава «Создание шаблона отчёта из кода».

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

Этот метод позволяет зарегистрировать пользовательскую функцию, которую впоследствии можно вызвать из отчёта. Функция вызывается посредством генерации события OnUserFunction объектом TfrxReport. Аргумент **FunctionDefinition** должен соответствовать активному формату скриптового языка. (Смотрите следующую главу для информации о поддерживаемых скриптовых языках). Следующая таблица приводит несколько примеров аргумента **FunctionDefinition:**

Язык	Пример аргумента FunctuionDefinition
Pascal	`function SpellValue(Value: String): String` `function AnotherFn(First: Integer; Second: Extended): Extended`
C++	`int MyExternalfunction(int idx, char * str)` `double CalcSomething(double x, double y, double z)`

Аргумент **Category** позволяет отнести функцию к логической группе. Группа не влияет на свойства функции, но используются дизайнером для удобной группировки функций в окне «Функции» дизайнера отчётов.

Аргумент **Description** определяет пользовательское описание функции. Этот аргумент не влияет на свойства функции, но показывается дизайнером отчёта.

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

Этот метод сохраняет подготовленный отчёт в COM или .NET поток данных (Stream). Метод может быть использовать для сохранения подготовленного отчёт в базу данных, передачу отчёта по локальной сети или передачу отчёта в любой другой объект, который поддерживает потоки (streams). Аргумент **Stream** задаётся посредством интерфейса IUnknown. Если аргумент Stream не является COM или .NET потоком данных, то метод вернёт соответствующую ошибку.

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

Этот метод возвращает версию FR Studio. Аргумент принимает значение версии в виде строки, содержащей старший и младший номера версии, разделённые точкой.

В будущих версиях FR Studio планируется введение также номера билда.

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

Этот метод возвращает страницу шаблона отчёта по её номеру. Аргумент **Index** определяет номер возвращаемой страницы. Аргумент **Value** принимает значение интерфейса **IfrxPage*** запрошенной страницы.

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

тот метод возвращает количество страниц шаблона в отчёте.

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

Этот метод производит поиск объекта шаблона отчёта по его уникальному имени. Аргумент **ObjectName** определяет имя искомого объекта. Строка, определяющая имя объекта – регистрозависимая. Аргумент **Obj** принимает значение интерфейса **IfrxComponent*** найденного объекта. Описание **IfrxComponent***интерфейса смотрите в разделе «Иерархия объектов».

Замечание: метод **FindObject** добавлен в версии 3.19, поэтому старые примеры, поставляемые в составе FR Studio, используют устаревшую методику поиска. Использование метода **FindObject** объекта **TfrxReport** наиболее предпочтительно..

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

Этот метод возвращает интерфейс на объект-хранилище страниц подготовленного отчёта. Это хранилище содержит все страницы, сгенерированные методом **PreapreReport**. Смотрите описание интерфейса **IfrxCustomPreviewPages*** в главе «Свойства объекта TfrxReport»

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

Этот метод снимет привязку всех датасетов от объекта TfrxReport.

Свойства объекта TfrxReport

Объект TfrxReport обладает следующим свойствами. Далее описываются непосредственно свойства объекта, затем подробное описание каждого набора свойств.

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

Набор свойств, относящихся к движку FastReport.

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

Набор свойств, управляющих окном предварительного просмотра.

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

Набор свойств, управляющих печатью отчёта на принтер. Некоторые свойства из этого набора влияют также на экспорт отчета.

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

Набор свойств, относящихся непосредственно к шаблону отчёта.

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

Набор свойств, управляющих ресурсами движка..

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

Это свойство определяет язык скрипта, используемого в отчёте. Формат этого свойства – строковое значение. В таблице приводится список допустимых значений для этого свойства.

Script language
PascalScript
C++Script
BasicScript
JScript

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

Это свойство предоставляет доступ к тексту скрипта отчёта.

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

Это свойство содержит текстовые строки ошибок, которые возникли при работе FR Studio. Свойство доступно только для чтения.

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

Это свойство позволяет указать дескриптор главного окна приложения. Свойство доступно только для записи. Используйте это свойство, для того чтобы указать главное окно для дизайнера и окна предварительного просмотра. Т.е. при помощи этого свойства дизайнер или окно предварительного просмотра можно сделать частью прикладной программы, таким образом, эти окна минимизируются и восстанавливаются вместе с приложением.

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

Это свойство запрещает отображение диалоговых окон отчёта. Использование этого свойства имеет смысл для пакетной обработки отчётов, т.е. в автоматическом режиме. В этом случае диалоговые окна отображаться не будут и параметры отчёта, которые обычно задаются пользователем в диалоговом окне, примут значения по умолчанию. В данном случае параметры, если необходимо, могут быть заданы из прикладной программы.

Описание набора свойств движка генератора отчётов

```
interface IfrxEngineOptions : IUnknown
{
HRESULT _stdcall ConvertNulls([out, retval] VARIANT_BOOL* Value);
HRESULT _stdcall ConvertNulls([in] VARIANT_BOOL Value);
```

Это свойство отвечает за преобразование Null значения поля базы данных в значение 0, «false» или пустую строку, в зависимости от типа поля данных.

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

Это свойство управляет режимом двухпроходного построения отчёта.

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

Это свойство задаёт максимальный размер памяти (в Мегабайтах) для кэша страниц отчёта. Это свойство учитывается, в случае если свойство UseFileCashe установлено в значение «истина». Если в момент построения отчёта он занимает больше памяти, чем определено этим свойством, то происходит кэширование во временный файл. Следует отметить, что это свойство задаёт только приблизительное значение размера памяти.

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

Это свойство определяет, нужно ли печатать пустые отчёты, т.е. отчёты, которые не содержать строк данных.

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

Это свойство определяет «тихий» режим. В этом режиме все ошибки, возникающие при работе генератора отчётов, не приводят к появлению диалоговых окон, а только сохраняются в свойстве Errors. Это свойство может оказаться полезным в режиме пакетного построения отчётов или при построении отчётов в среде ASP.NET.

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

Это свойство определяет местоположение папки, в которой будут создаваться временные файлы.

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

Это свойство управляет режимом кэширования страниц подготавливаемого отчёта во внешний файл. Это свойство может быть полезно при построении больших отчётов на машинах с ограниченным размером памяти. Смотрите описание свойства "MaxMemSize".

```
};
```

Описание набора свойств окна предварительного просмотра

```
interface IfrxPreviewOptions : IUnknown
{

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

Это свойство разрешает или запрещает редактирование сгенерированного отчёта.

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

Это свойство определяет набор кнопок, которые будут доступны в окне предварительного просмотра. Значение параметра Value есть набор, состоящий из комбинации нижеприведённых параметров:

```
enum {

    pb_Print = 1,            //  кнопка печати отчёта
    pb_Load = 2,             //  кнопка загрузки подготовленного отчёта
    pb_Save = 4,             //  кнопка сохранения подготовленного отчёта
    pb_Export = 8,           //  кнопка экспорта отчётов
    pb_Zoom = 16,            //  кнопки масштабирования
    pb_Find = 32,            //  кнопка поиска
    pb_Outline = 64,         //  кнопка переключения схемы документа
    pb_PageSetup = 128,      //  кнопка параметров страницы
    pb_Tools = 256,          //  кнопки инструментов
    pb_Edit = 512,           //  кнопка редактирования
    pb_Navigator = 1024,     //  кнопки навигатора
    pb_ExportQuick = 2048    //  кнопка быстрого экспорта

} frxPreviewButtons;
```

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

Это свойство управляет режимом двойной буферизации для окна предварительного просмотра. Это режим используется по умолчанию. Если он включен, окно предварительного просмотра не будет мерцать при перерисовке, однако, этот режим считается более медленным.

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

Это свойство определяет, является ли окно предварительного просмотра максимизированным на всю рабочую область экрана.

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

Это свойство определяет, является ли окно предварительного просмотра дочерним окном в MDI приложении.

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

Это свойство определяет, является ли окно предварительного просмотра модальным.

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

Это свойство определяет, является ли окно структуры документа расширенным.

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

Это свойство управляет видимостью окна структуры документа.

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

Это свойство управляет шириной окна структуры документа.

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

Это свойство определяет необходимость отрисовки надписей на кнопках окна предварительного просмотра. При установке этого свойство в значение «истина», Вам необходимо ограничить количество кнопок при помощи свойства Buttons, вследствие того, что ширина всех кнопок с включенными надписями гораздо больше ширины экрана.

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

Это свойство управляет масштабом отображения страниц подготовленного отчёта.

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

Это свойство позволяет выбрать масштаб страниц подготовленного отчёта из набора заданных масштабов:

```
zmDefault    - для масштабирования будет использоваться свойство Zoom;
zmWholePage  - Использовать масштаб «Одна страница на экран»;
zmPageWidth  - Использовать масштаб «По ширине страницы»;
zmManyPages  - Использовать масштаб «Две страницы на экран»;

};
```

Описание набора свойств печати `отчёта`

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

Это свойство задаёт количество копий при печати отчёта на принтере.

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

Это свойство управляет сортировкой копий страниц при печати отчёт на принтер.

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

Это свойство определяет номера страниц для печати и экспорта. Номера страниц задаются в виде строки. Формат строки аналогичен формату списка страниц печати в Microsoft Word, к примеру: "1,3,5-12,17-".
Пустая строка является признаком печати/экспорта всех страниц.

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

Это свойство определяет имя принтера для печати. Пустая строка означает использование принтера, используемого в системе по умолчанию.

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

Это свойство определяет, какие страницы должны быть напечатаны. Параметра Value может принимать одно из следующих значений:

```
    ppAll – печатать все страницы
    ppOdd – печатать только нечётные страниц
    ppEven – печатать только чётные страницы
```

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

Это свойство определяет, должен ли быть показан диалог выбора параметров печати перед печатью. По умолчанию, это свойство установлено в значение «false», таким образом, печать отчёта из прикладной программы посредство метода PrintReport не приведет к появлению окна диалога печати. Однако, при печати из окна предварительного просмотра, диалог показывается всегда.

```
};
```

Описание набора свойств шаблон отчёта

```
interface IfrxReportOptions : IUnknown
{

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

Это свойство определяет автора шаблона отчёта.

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

Это свойство определяет, необходимо ли сохранять шаблоны отчётов и подготовленные отчёт в формате сжатия GZIP.

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

Это свойство определяет имя, по которому выбирается строка соединения к базе данных. Это соединение используется объектами TfrxADOTable и TfrxADOQuery в том случае, если отчёт не содержит объекта TfrxADOConnection. Такое подключение к базе данных называется соединение по умолчанию (DefaultConnection). Соответствие между именем соединения и строкой подключения к базе данных определено в системном реестре. Подробнее об этом смотрите в разделе «Работа с объектом TfrxReport» глава «Использование соединения по умолчанию».

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

Это свойство управляет датой создания шаблона отчёта.

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

Это свойство управляет кратким описанием отчёта.

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

Это свойство управляет именем шаблона отчёта.

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

Это свойство содержит дату последней модификации шаблона отчёта.

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

Это поле управляет паролем отчёта. Если это поле не пустое, то при открытии отчёта будет запрошен пароль.

```
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);
```

Эти свойства определяют номера версий шаблона отчёта. Эти свойства не заполняются автоматически и логически не связаны с версией FastReport.

```
};
```

Описание набора свойств и методов ресурсов движка.

```
interface IfrxResources : IUnknown
{

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

Это свойство определяет путь к файлу помощи, который будет показан при нажатии клавиши F1 окне дизайнера.

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

Показать окно помощи. Будет отображен файл, указанный в свойстве HelpFile. Если файл помощи не найден, то операция игнорируется без сообщения об ошибке.

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

Этот метод позволяет получить строковое описание ресурса, заданное параметром ID. Для корректной работы этого метода необходимо предварительно загрузить ресурсы при помощи метода LoadLanguageResourcesFromFile.

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

Это метод используется для выбора и загрузки файла, содержащего ресурсы для пользовательского интерфейса дизайнера и окна предварительного просмотра отчётов. Аргумент FileName представляет собой полный путь к файлу, содержащему языковые ресурсы. Файлы ресурсов имеют расширение «.frc». Каждый файл ресурсов содержит ресурсы для определённого национального языка. По умолчанию, FastReport использует ресурсы для английского языка.

```
};
```

IfrxCustomPreviewPages

Этот интерфейс управляет хранилищем, которое содержит страницы сгенерированного отчёта. Используйте этот интерфейс осторожно, поскольку он считается низкоуровневым интерфейсом.

```
interface IfrxCustomPreviewPages : IDispatch {

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

Внутренний метод. Не используйте его.

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

Добавляет страницу к хранилищу подготовленных страниц.

```
    HRESULT _stdcall AddEmptyPage([in] long Index);
```

Добавляет пустую страницу, в позицию, заданную параметром 'Index'.

```
    HRESULT _stdcall DeletePage([in] long Index);
```

Удаляет станицу, в позиции, заданной параметром 'Index'.

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

Возвращает число сгенерированных страниц.

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


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

Управляет активной страницей, которая в настоящий момент видима в окне предварительного просмотра.

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

Возвращает в параметре Value интерфейс IfrxReportPage* на объект страницы подготовленного отчёта с номером, заданным параметром Index. Интерфейс объекта страницы описан в разделе «Иерархия объектов»

```
};
```

События, генерируемые объектом TfrxReport

Во время геренации отчета, объект TfrxReport инициирует различные события. Помимо этого, некоторые события могут генерироватся окном предварительного просмотра отчёта.

Все события объекта TfrxReport реализованы посредством диспетчеров IfrxReportEvents и IfrxReportEventDispatcher. Эти интерфейсы одинаковые, за исключением того, что первый использует для обработки событий в VC++, а второй для .NET делегатов.

События занимают очень важное место в генерации отчётов. Например, динамическое назначение картинок объекту TfrxPictureView или ввод данных в объект TfrxCrossView, возможны только в событии OnBeforerPrint. Событие OnAfterPrint целесообразно использовать для освобождения ресурсов, которые были выделены в событии OnBeforePrint.

Многие события имеют параметр Sender, который имеет тип IfrxComponent. Интерфейс IfrxComponent – базовый интерфейс для всех объектов отчёта. Используя этот интерфейс, можно узнать имя и тип объекта, являющегося источником события. Эти данные необходимы для корректной обработки события. Подробное описание интерфейса IfrxComponent и от него наследованных объектов смотрите в разделе «Иерархия объектов».

```
interface IfrxReportEvents : IDispatch
{

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

Это событие генерируется после обработки каждого объекта. Аргумент Sender указывает на объект, который инициировал событие.

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

Это событие генерируется перед обработкой каждого объекта. Аргумент Sender указывает на объект, который инициировал событие.

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

Это событие генерируется в момент, когда пользователь щелкает мышью на каком либо объекте в окне предварительного просмотра сгенерированного отчёта. В данном случае аргумент Sender указывает на объект, на котором произведён щелчок. Аргумент Button определяет, какая кнопка мыши была нажата.

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

Это событие генерируется в момент генерации отчёта, когда встроенный парсер определяет вызов пользовательской функции, ранее зарегистрированной при помощи метода "AddFunction" объекта TfrxReport. Смотрите пример регистрации и использования пользовательских функций в разделе «Работа с объектом TfrxReport» в главе «Вызов внешних функций».

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

Это событие инициируется непосредственно перед генерацией отчёта.

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

Это событие инициируется непосредственно после окончания генерации отчёта.

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

Это событие генерируется после печати отчета на принтере.

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

Это событие периодически генерируется во время построения отчёта и сообщает информацию о прогрессе генерации.

```
HRESULT OnProgressStart();
```

Это событие информирует о начале генерации отчета.

```
HRESULT OnProgressStop();
```

Это событие информирует об окончании генерации отчёта.

```
};
```

Внимание: оригинальный "FastReport Delphi" поддерживает большее количество событий. Если Вы считаете, что в FR Studio не хватает какого либо события, существующего в FastReport – обращайтесь в support@fast-report.com. Не забудьте указать продукт «FastReport Studio».

Пользовательский набор данных - объект TfrxUserDataSet

Объект TfrxUserDataSet – используется для доступа к данным, не основанных на хранилище базы данных. Он служит для передачи данных из прикладной программы в генератор отчётов. Объект поддерживает несколько событий для навигации и доступа к данным. С помощью объекта TfrxUserDataSet Вы можете создавать отчёты, которые принимают данные из массивов, таблиц, внешних файлов и других источников данных. Для использования этого объекта, программист обязан обеспечить реализацию функций навигации по таким источникам данных (смотрите описание событий объекта TfrxUserDataSet). Помимо этого, объект TfrxUserDataSet используется для реализации обёртки классов ADO.NET. Этот объект не имеет методов, но имеет несколько свойств и событий, которые описаны в этой главе. Этот объект также поддерживает интерфейс IfrxDataSet, описанный в разделе «Иерархия объектов».

Замечания для программистов, использующих ADO.NET: Терминология FastReport'а несколько отличается от терминологии, принятой в среде ADO.NET. По терминологии ADO.NET, DataSet представляет собой хранилище таблиц, Views и Relations, однако по терминологии FastReport, DataSet – это аналог ADO.NET таблиц. Пожалуйста, учтите это различие и избежите путаницы в терминологии.

Свойства пользовательского набора данных

Объект TfrxUserDataSet поддерживает следующие свойства:

```
interface IfrxUserDataSet : IDispatch
{

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

Список наименований полей пользовательского набора данных. Каждое поле представлено в виде строки, отделенной от следующей символами перевода строки и возврата каретки (CR/LF). Это свойство можно представить как аналог поля в записи базы данных.

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

Это свойство определяет уникальное имя пользовательского набора данных. Это имя будет отображаться в дизайнере отчётов. Следует избегать использования объектов с одинаковыми именами, в противном случае возможны ошибки в работе генератора отчётов.

Если Вы динамически создаёте различные наборы данных в среде .NET, то рекомендуется принудительно запускать сборщик мусора после освобождения объекта или выбирать уникальное имя для каждого набора даже после его освобождения. Эта особенность связана с тем, что на самом деле объект разрушается значительно позже после его освобождения, что приводит к одновременному существованию нескольких объектов с одинаковым именем. Поскольку FastReport оперирует наборами данных по их имени, возможны нежелательные эффекты при построении отчётов. Данная особенность не проявляется в non-managed средах.

```
};
```

События пользовательского набора данных

Следующие события определены для объекта TfrxUserDataSet. Для корректной работы объекта эти события обязательно должны быть обработаны в прикладной программе. Назначение этих событий – навигация по источнику данных и передача данных в отчёт.

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

Это событие служит для передачи данных в отчёт. Аргумент VarName определяет имя поля, для которого запрошены данные. Аргумент Value принимает запрошенное значение, возвращаемое обработчиком событий.

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

Обработчик события должен установить значение аргумента IsEOF в «истина», в случае, если достигнут конец набора данных, то есть если все данные обработаны. В противном случае значение IsEOF должно быть установлено в «ложь».

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

Обработчик этого события устанавливает курсор в начало набора данных, например, к первому элементу массива, к первой строке файла и т.д.

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

Обработчик этого события устанавливает курсор на следующую запись.

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

Обработчик этого события устанавливает курсор на предыдущую запись.
};

Встроенные компоненты ADO

Компоненты ADO это набор встроенных объектов, которые предоставляют доступ к движку ADO. FT Studio поддерживает три таких компонента: «TfrxADODatabase», «TfrxADOTable» и «TfrxADOQuery».

Существует два пути использования ADO объектов. Первый, и наиболее часто используемый способ, – внедрение ADO объектов в отчёт. То есть ADO объекты создаются в дизайнере отчётов и сохраняются в шаблоне отчёта. Для получения доступа к этим объектам из прикладной программы, необходимо загрузить отчёт при помощи метода TfrxReport.LoadReport, а затем произвести поиск этого объекта по его имени при помощи метода TfrxReport.FindObject.

Второй способ – динамическое создание ADO объектов в прикладной программе. В этом случае потребуется назначить созданные объекты TfrxADOTable и TfrxADOQuery шаблону отчёта при помощи метода TfrxReport.SelectDataset. Помимо этого, Вам потребуется назначить эти объекты бэндам, используемым в шаблоне отчёта. Как это сделать смотрите в разделе «Иерархия объектов».

Объект TfrxADODatabase

Объект TfrxADOConnection предоставляет соединение к базе данных посредством движка ADO, используйте его для подключения к базам данных. Один объект TfrxADOConnection может использоваться несколькими объектами таблиц TfrxADOTable и несколькими объектами запросов TfrxADOQuery. Взаимосвязь между объектами TfrxADOTable и TfrxADOQuery и объектом TfrxADOConnection осуществляется посредством свойства DataBase, которое имеется у TfrxADOTable и TfrxADOQuery.

```
interface IfrxADODatabase : IDispatch {
```

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

Это свойство управляет строкой соединения, которое описывает используемую базу данных. Свойство ConnectionString описывает тип базы данных, путь к базе данных и параметры соединения с базой данных. Для подробного описания формата ConnectionString используйте MSDN или сгенерируйте строку при помощи ADO Connection Wizard.

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

Это свойство управляет выводом окна ввода параметров подключения к базе данных. Это окно используется для ввода пользователем имени и пароля.

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

Определяет активность соединения к базе данных. Установка этого свойства в значение «Истина» производит подключение к базе данных. Установка свойства в значение «Ложь» производит разрыв соединения с базой данных. Также, пользовательская программа может определить текущий статус подключения к базе данных посредством проверки этого свойства. Соответственно, значение «Истина» означает, что подключение к базе данных активно. Значение «Ложь» - подключение неактивно.

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

Определяет временной интервал, в течение которого должно установиться соединение с базой данных. При превышении интервала соединение не устанавливается.

```
};
```

Объект TfrxADOTable

Объект TfrxADOTable служит для доступа к таблицам баз данных ADO. Этот объект служит источником данных для построения отчётов. Методы и свойства объекта предоставляются посредством интерфейса IfrxADOTable, который является наследником от IfrxDataSet.

```
interface IfrxADOTable : IDispatch {

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

Это свойство управляет привязкой таблицы к базе данных, т.е. к объекту TfrxADODatabase. Таблица не может быть использована без привязки к базе данных.

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

Свойство IndexName определяет текущий активный индекс. Используйте это свойство для активации индекса, что позволит изменить текущий порядок записей в таблице. Для этого необходимо присвоить этому свойству строку, содержащую имя необходимого индекса.

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

Свойство TableName определяет таблицу базы данных, которая отображается в объект TfrxADOTable.

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

Свойство Name определяет символическое имя таблицы. Это имя будет отображаться в дизайнере. Помимо этого, операции поиска объекта будут производиться по этому имени.

```
};
```

Объект TfrxADOQuery

Объект TfrxADOQuery предоставляет средство для исполнения SQL запросов на стороне базы данных ADO. Результаты этих запросов могут отображаться на страницах отчёта. Методы и свойства объекта предоставляются посредством интерфейса IfrxADOQuery, который является наследником от IfrxDataSet.

```
interface IfrxADOQuery : IDispatch {

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

Это свойство управляет привязкой запроса к базе данных, т.е. к объекту TfrxADODatabase. Запрос не может быть выполнен и использован без привязки к базе данных.

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

Свойство Query управляет SQL запросом, который будет использоваться для выборки данных из базы данных.

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

Свойство Name определяет символическое имя таблицы. Это имя будет отображаться в дизайнере. Помимо этого, операции поиска объекта будут производиться по этому имени.

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

Свойство CommandTimeout определяет значение максимального интервала времени, в течение которого должен выполниться SQL запрос. При превышении интервала, возвращается ошибка.

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

Метод ParamByName позволяет передавать дополнительные параметры в SQL запрос.

};

ActiveX объекты

FR Studio предоставляет два ActiveX объекта – окно предварительного просмотра отчёта и форму предварительного просмотра отчёта. Эти объекты отличаются лишь набором предоставляемых возможностей.

TfrxPreviewX

Окно предварительного просмотра отчёта более простой объект – он не содержит графических элементов управления просмотром: масштабирования, навигации, экспорта и других контролов. Этот объект предоставляет два интерфейса – IfrxpreviewX и IfrxPreviewEvents.

```
dispinterface IfrxPreviewX {

        void _stdcall Init();
```

Производит начальную инициализацию объекта.

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

Эти методы блокируют и разблокируют отрисовку ActiveX объекта.

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

Этот метод добавляет одну пустую страницу к подготовленному отчёту.

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

Удаляет текущую страницу в подготовленном отчёте.

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

Печатает подготовленный отчёт на выбранном принтере. Свойства печати могут быть заданы посредством свойства PrintOptions интерфейса IfrxReport.

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

Загружает подготовленный отчёт из файла, чьё имя задано параметром FileName.

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

Сохраняет подготовленный отчёт в файл, чьё имя задано параметром FileName.

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

Запускает подготовленный отчёт во встроенном редакторе.

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

Эти методы позволяют производить навигацию в подготовленном отчёте.

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

Этот метод отображает диалог установки свойств страницы отчёта.

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

Этот метод отображает диалоговое окно поиска.

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

Очищает подготовленный отчёт.

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

Этот метод устанавливает текущую позицию в окне подготовленного отчёта. Параметр PageN указывает номер страницы, параметр Top указывает на смещение относительно верхнего края страницы.

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

Эти методы позволяют отобразить/скрыть окно с сообщением, заданным параметром s, сверху окна просмотра подготовленного отчёта.

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

Этот метод позволяет моделировать мышиное колесо.

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

Этот метод возвращает число страниц в подготовленном отчёте.

```
TxfrxPreviewTool _stdcall Tool();
```

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

Controls preview tool

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

Эти свойства управляют масштабом отображения подготовленного отчёта.

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

Это свойство позволяет установить один из нескольких режимов предварительного просмотра:

```
enum {
    zm_Default = 0,
    zm_WholePage = 1,
    zm_PageWidth = 2,
    zm_ManyPages = 3
} frxZoomMode;
```

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

Управляет видимостью окна схемы документа.

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

Управляет шириной окна схемы документа

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

Управляет режимом двойной буферизации.

```
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);
```

Вышеприведенные методы и свойства относятся к методам ActiveX протокола.

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

Свойство, которое привязывает отчёт к окну предварительного просмотра подготовленного отчёта.

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

Позволяет загружать подготовленный отчёт из COM или .NET потоков данных. (Stream)

};

TfrxActivePreviewForm

Объект TfrxActivePreviewForm отображает форму с подготовленным отчётом. Эта форма включает в себя продвинутый пользовательский интерфейс с такими контроллами как масштабирование, экспорты, кнопки навигации и другие. Объект TfrxPreviewForm предоставляет два интерфейса – IfrxActivePreviewForm и IfrxActivePreviewFormEvents.

```
dispinterface IfrxActivePreviewForm {

        VARIANT_BOOL _stdcall AutoScroll();
        void _stdcall AutoScroll([in] VARIANT_BOOL rhs);
        VARIANT_BOOL _stdcall AutoSize();
        void _stdcall AutoSize([in] VARIANT_BOOL rhs);
        TxActiveFormBorderStyle _stdcall AxBorderStyle();
        void _stdcall AxBorderStyle([in] TxActiveFormBorderStyle rhs);
        BSTR _stdcall Caption();
        void _stdcall Caption([in] BSTR rhs);
        OLE_COLOR _stdcall Color();
        void _stdcall Color([in] OLE_COLOR rhs);
        IFontDisp* _stdcall Font();
        void _stdcall Font([in] IFontDisp* rhs);
        void _stdcall Font([in] IFontDisp** rhs);
        VARIANT_BOOL _stdcall KeyPreview();
        void _stdcall KeyPreview([in] VARIANT_BOOL rhs);
        long _stdcall PixelsPerInch();
        void _stdcall PixelsPerInch([in] long rhs);
        TxPrintScale _stdcall PrintScale();
        void _stdcall PrintScale([in] TxPrintScale rhs);
        VARIANT_BOOL _stdcall Scaled();
        void _stdcall Scaled([in] VARIANT_BOOL rhs);
```

Методы и свойства, описанные выше, относятся к реализации ActiveX протокола.

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

Загружает подготовленный отчёт из файла. Параметр Value указывает на имя загружаемого файла.

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

Возвращает интерфейс объекта отчёта.

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

Привязывает отчёт к форме предварительного просмотра.

};

Объект TfrxGzipCompressor

Объект TfrxGzipCompressor служить для сжатия и распаковки потоков данных. Этот объект поддерживает .COM потоки и .NET потоки. Объект предоставляет интерфейс IfrxCustomCompressor, который включает в себя два метода.

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

Этот метод сжимает поток. Источник данных задается параметром InputStream. Результирующий поток указывается параметром OutputStream. Параметр CompressionRate указывает степень сжатия данных. Параметр FileName указывает имя файла, которое будет сохранено в архиве, как оригинальное имя сжимаемого документа.

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

Этот метод распаковывает входной поток в файл, с именем, указанным аргументом FileName при компрессии.

Объект TfrxReportClient object

Объект TfrxReportClient используется для связи с FastReport Server. Этот объект позволяет установить соединение, запросить подготовку отчёта и загрузить подготовленный отчёт в объект TfrxReport. Объект TfrxReportClient предоставляет интерфейс со следующими методами и свойствами:

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

Это свойство возвращает интерфейс на объект соединения с сервером, который контролирует все свойства соединения между клиентом и сервером. Описание интерфейса IfrxReportServerConnection приведено ниже.

```
void PrepareReport();
```

Выполняет генерацию отчёта на стороне сервера и загружает подготовленный отчёт на сторону клиента.

```
void ShowReport();
```

Отображает отчёт, подготовленный на сервере, в окне просмотра на локальном компьютере.

```
BSTR ReportName();
```

Возвращает имя подготовленного отчёта.

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

Устанавливает имя отчёта.

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

Запрос значения переменной с сервера.

Перед установкой соединения между клиентом и сервером, на стороне клиента должны быть заданы несколько свойств. Эти свойства управляются посредством интерфейса IfrxReportServerConnection. Данный интерфейс может быть запрошен у объекта TfrxReportClient посредством метода ReportServerConnection().

```
interface IfrxReportServerConnection : IDispatch {

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

Управляет использование компрессии при передаче данных между клиентом и сервером.

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

Это свойство управляет именем хоста FR сервера. Значение Value может быть как доменным именем, таки IP адресом. Это свойство необходимо для установки соединения с сервером.

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

Это свойство управляет именем пользователя при авторизации процесса соединения с сервером.

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

Это свойство управляет включением/выключением проверки целостности данных при помощи MD5 контрольной суммы для данных, передаваемых между клиентом и сервером.

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

Это свойство управляет паролем при авторизации процесса соединения с сервером.

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

Это свойство управляет номером TCP порта для соединения с сервром.

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

В случае использования прокси сервера, это свойство управляет номером TCP порта прокси сервера.

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

Это свойство управляет именем хоста прокси сервера. Также может быть зада IP адрес. Пустая строка отключает использование прокси сервера. По умолчанию прокси сервер не используется.

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

Это свойство управляет числом повторов посылки данных, в случае коммуникационных проблем с сервером.

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

Это свойство управляет временным интервалом, используемым для повторной передачи данных, в случае коммуникационных проблем. Время задаётся в секундах.

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

Это свойство управляет временным интервалом для управляющих операций.

```
};
```

Использование объекта TfrxReport

Как уже сказано выше, TfrxReport – основной объект генератора отчётов. Его использование покрывает большую часть стандартных возможностей генератора отчётов. Поэтому данный раздел содержит описание основных принципов работы с отчётом.

Конфигурирование и установка среды

Способ создания объекта отчёта определяется языком программирования. Далее следуют описания для Visual C++, C# и Visual Basic .NET.

Visual C++

Наиболее удобный способ работы с FR Studio – использование ATL. Нижеприведенный код делает доступными классы FastReport в проекте Visual Studio. Использование встроенной директивы import позволяет автоматически сгенерировать классы FastReport на основе поставляемой динамической библиотеки. Как видно из примера, существует две формы привязки FastReport к проекту – путём задания пути к динамической библиотеке и посредством задания libid. Первый способ более портабелен, поскольку поддерживается в шестой версии Visual C++, второй способ более удобен, поскольку при его использовании нет необходимости знать место установки dll.

```
#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

Для использования FR Studio в Managed средах необходимо добавить reference к библиотеке FastReport3.dll. Для этого необходимо использовать «Add reference» диалог, возникающий при левом клике на пункте Reference в окне «Solution explorer». Выберите вкладку COM в окне « Add reference», найдите компонент «FastReport3 report generator» и выберите его. Нажмите кнопку Ok.

Создание экземпляра объекта TfrxReport

Visual C++

Используйте код, приведённый выше, для генерации заголовочных файлов и установки соответствующего пространства имён. Затем используйте следующую строку для создания объекта отчёта:

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

Обратите внимание на использование AutoPtr функциональности.

C#.NET

Воспользуйтесь шагами, описанными в предыдущей части. Затем используйте следующий код для создания объекта отчёта:

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

Visual Basic.NET

Воспользуйтесь шагами, описанными в предыдущей главе. Затем, используйте следующий код для создания объекта отчёта:

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

Загрузка и сохранение отчёта

Последующие примеры практически идентичны для всех платформ. Небольшое исключение составляет C++ - используйте указатели для C++.

Visual C++

```
// Загружает отчёт из файла "1.fr3"
pReport->LoadReportFromFile("c:\1.fr3");
// Сохраняет отчёт в файл "1.fr3"
pReport->SaveReportToFile("c:\2.fr3");
```

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

```
// Загружает отчёт из файла "1.fr3"
report.LoadReportFromFile("c:\1.fr3");
// Сохраняет отчёт в файл "1.fr3"
report.SaveReportToFile("c:\2.fr3");
```

Дизайн отчета

Шаблон отчёта может быть построен двумя способами. В большинстве случаев отчёт строится прикладным программистом или пользователем при помощи встроенного дизайнера отчётов. Вызов встроенного дизайнера отчётов осуществляется посредством метода DesignReport, который предоставляет объект TfrxReport. Дизайнер открывает

шаблон отчёта, который был ранее загружен при помощи метода LoadReportFromFile или LoadreportFromStream. Если перед вызовом DesignReport шаблон отчета не был загружен, в этом случае дизайнер открывает пустой шаблон отчёта.

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

Другим способом создания шаблона отчёта является его динамическое построение из прикладной программы. Примеры динамического построения отчётов приведены ниже.

Построение отчёта

Используйте один из двух способов построения отчётов:

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

В большинстве случаев, использование ShowReport более удобно – этот метод начинает отображать отчёт ещё в процессе его построения.

Параметр **ClearLastReport** метода PrepreReport удобен при добавлении нового подготовленного отчёта к уже существующему. Такие отчёты называются композитными отчётами. Обратите внимание, что метод PrepareReport не отображает подготовленный отчёт, в отличие от метода ShowReport.

Предварительный просмотр отчёта

Предварительный просмотр отчёта возможен при помощи использования одной из двух методов. Первый – ShowReport описан выше. Второй способ – использование метода ShowPreparedReport. Во втором случае не происходит построение отчёта, а отображается ранее подготовленный отчёт. Таким образом, отчёт должен быть подготовлен заранее при помощи метода PrepareReport (описан выше) или загружен при помощи методов LoadPreparedReportFromFile / LoadPrepareReportFromStream.

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

Вышеприведенный пример осуществляет построение отчёта, затем, если не было ошибок при построении отчёта, отображает его в окне предварительного просмотра. Построение большого отчёта может занять длительное время, поэтому использование асинхронного метода ShowReport может быть более предпочтительно. Различные параметры предварительного просмотра могут быть заданы при помощи свойства PreviewOptions объекта TfrxReport.

Печать отчёта

В большинстве случаев печать осуществляется из окна предварительного просмотра при помощи кнопки Print. Для печати отчёта из прикладной программы используйте метод Print объекта TfrxReport. Пример:

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

При вызове этого метода появится диалоговое окно печати, в котором можно выбрать различные параметры печати. Установка параметров печати, а также запрещение вывода диалога для пакетного режима работы осуществляется при помощи свойства PrintOptions объекта TfrxReport.

Загрузка и сохранение подготовленного отчёта

Подготовленный отчёт может быть сохранён и загружен из окна предварительного просмотра подготовленного отчёта. Также, загрузка и сохранение подготовленного отчёта может быть осуществлена прикладной программой при помощи следующих методов?

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

Файл, содержащий подготовленный отчёт, по умолчанию имеет расширение «.FP3»

Обратите внимание, что после загрузки подготовленного отчёта, его просмотр осуществляется при помощи метода ShowPreparedReport.

Экспорт подготовленного отчёта

FastReport имеет мощные возможности экспорта – он поддерживает множество широко распространённых форматов данных, таких как Adobe PDF, Rich Text Format, битмаповые картинки, простой текст и другие форматы.

Замечание: Начиная с билда 3.21.50, свойство PageNumbers интерфейса IfrxPrintOptions также влияет на все виды экспорта.

Использование встроенных экспортов несколько отличается при использовании C++ и в managed средах.

Visual C++

Для использования возможностей экспорта из C++ приложения требуется запросить интерфейс IfrxBuiltinExports. Этот интерфейс должен быть получен у объекта TfrxReport. Следующий код иллюстрирует сказанное:

```
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

Экспорт в managed средах выглядит несколько проще:

```
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);
```

Обратите внимание, что экспорт в из ASP.NET приложения может быть недоступен в связи с правами доступа на рабочую директорию. Таким образом, необходимо учесть этот фактор и производить экспорт в директорию, доступную на запись ASP.NET приложению.

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

Экспортирует подготовленный отчет в формат PDF.

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

Экспортирует подготовленный отчет в XML формат.

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

Экспортирует подготовленный отчет в RTF формат.

```
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);
```

Экспортирует подготовленный отчет в HTML формат. Нижеприведенная таблица содержит описание параметров HTML экспорта:

Имя параметра	Тип	Описание
FileName	String	Имя файла(ов) для экспорта
Pictures	Boolean	Разрешить или запретить экспорт картинок
FixedWidth	Boolean	Если «истина», то экспортируемые страницы имеют фиксированную ширину.
Multipage	Boolean	Экспортировать каждую страницу отчёта в отдельный HTML файл.
Navigator	Boolean	Если «истина», то экспортировать дополнительную страницу навигации.
PicsInSameFolder	Boolean	Если истина, то экспортировать картинку в ту же папку что и отчёт
Background	Boolean	Если истина, то экспортировать фоновую картинку

```
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);
```

Экспортирует подготовленный отчет в формат MS Excel. Этот тип экспорта требует наличие установленного на компьютере продукта Microsoft Excel.

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

Экспортирует подготовленный отчет в картинку формата BMP.

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

Экспортирует подготовленный отчет в картинку формата JPEG.

```
HRESULT ExportToTIFF(
```

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

Экспортирует подготовленный отчет в формат TIFF.

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

Экспортирует подготовленный отчет в простой текстовый формат. При этом теряется графическое оформление отчёта.

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

Экспортирует подготовленный отчет в формат CSV.

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

Экспортирует подготовленный отчет в картинку формата GIF.

```
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);
```

Посылает E-mail. Этот псевдоэкспорт позволяет посылать электронные письма с вложениями посредством протокола SMTP. Следующая таблица описывает параметры функции SenMail:

параметр	тип	описание	Пример
Server	String	URL или IP адрес SMTP сервера	"mail.fast-report.com"
Port	Integer	TCP порт SMTP сервера	25
User	String	Имя пользователя для SMTP авторизации	"mailbot"
Password	String	Пароль для SMTP авторизации	"Hjk344Rd"
From	String	E-mail адрес источника	"mailbot@fast-report.ru"
Recipient	String	E-mail адрес получателя	"boss@fast-report.com"
Subject	String	Тема письма	"Weekly report"
Text	String	Текст письма	"Hello Boss!\n\nBye!"
FileName	String	Имя файла вложения на диске	"C:\\Reports\\Sell.fp3"
AttachName	String	Имя файла вложения аттача	"NewSells.fp3"

Использование соединения к базе данных по умолчанию

Одной из возможностей формата шаблона отчёта FP3 – хранение имени соединения с базой данных в теле отчёта. Объекты TfrxADOTable и TfrxADOQuery используют соединение по умолчанию, в случае, если шаблон отчета не содержит ни одного объекта TfrxADODatabase. Для поддержания списка соединений по умолчанию, используйте дизайнер, поставляемый вместе со FR Studio. Для перехода к режиму редактирования соединений по умолчанию используйте подменю «Подключения» из меню «Вид». Для назначения соединения по умолчанию к редактируемому отчёту, используйте подменю «Данные» из меню «Отчёт». Обратите внимание, что данные операции должны проводиться во внешнем дизайнере отчётов, поскольку встроенный дизайнер отчётов не предоставляет такой функциональности.

Для динамического задания соединения по умолчанию из прикладной программы, используйте следующий код:

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

Внимание: Возможность соединения к базе данных по умолчанию не совмести с многопоточными приложениями. Для использования ADO соединения в многопоточных приложениях Вам необходимо явно использовать объект TfrxADODatabse, создав его в дизайнере отчётов или динамически. Использование соединения к базе данных по умолчанию в многопоточных приложения приведёт к ошибке доступа. Смотрите раздел «Использование в многопоточных приложениях» для дополнительной информации.

Внимание: При использовании соединения к базе данных по умолчанию, не производится проверка на доступность выбранной базы данных. Поэтому программа может вызвать ошибку при вызове метода PrepareReport().

Замечание: Описание строк соединения к базам данных по умолчанию находится в системном реестре. Путь к этим данным:

HKEY_LOCAL_MACHINE\SOFTWARE\Fast Reports\Connections key.

Каждое описание состоит из стоковых значений, где имя строки соответствует имени соединению по умолчанию, и данные строки содержать ADO connection string.

Использование в многопоточных приложениях

FastReport Studio использует так называемую Single Threaded Apartment Model. Это означает, что каждый программный поток может создать и использовать один или несколько объектов отчёта, но каждый объект отчета может быть использован только из одного программного потока.

Замечание: Мы обнаружили, что используемый код ADO не полностью поддерживает многопоточность. Нежелательные эффекты могут проявиться при использовании соединения к базе данных по умолчанию.

Построение композитного отчета (пакетная печать отчёта)

В некоторых случаях бывает необходимо организовать печать нескольких отчётов как один или отобразить несколько отчетов в одном окне предварительного просмотра. FastReport позволяет построить новый отчёт, в дополнение к уже существующему. Метод PrepareReport объекта TfrxReport имеет параметр ClearLastReport. Если этот параметр имеет значение «Ложь», то построение нового отчёта происходит как дополнение к уже построенному. Следующий пример показывает, как можно построить композитный отчёт:

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

Мы загружаем первый шаблон отчета и строим отчёт без его показа в окне предварительного просмотра. Затем мы загружаем следующий отчёт и строим его, установив параметр ClearLastReport в значение «Ложь». При этом второй отчёт добавляется к предыдущему. После этого мы отображаем композитный отчёт в окне предварительного просмотра.

Нумерация страниц в композитном отчёте

Вы можете использовать системные переменные «Page» «Page#» «TotalPages» и «TotalPages#» для отображения номера страницы или для отображения общего числа страниц в подготовленном отчёте. В композитных отчётах эти переменные работают следующим образом:

Page – номер страницы в текущем отчете
Page# - номер страницы в пакете (композитном отчёте)
TotalPages – общее число страниц в текущем отчёте (отчёт должен быть двухпроходным)
TotalPages# - общее число страниц в пакете (композитном отчёте)

Комбинация страниц в композитном отчёте

Страница шаблона отчёта имеет свойство PrintOnPreviousPage. Когда этому свойству присвоено значение «Истина», то происходит сращивание страниц встык, т.е. печать объекта со следующей страницы происходит на свободном месте предыдущей. В композитном отчёте это позволяет создавать новый отчёт на свободном пространстве последней страницы предыдущего отчёта. Для получения такого результата необходимо

установить свойство PrintOnPreviousPage в значение «Истина» для каждого первого шаблона страницы последующих отчётов.

Интерактивные отчёты

Существует два способа создания интерактивных отчетов. Первый – использование встроенного скрипта и перехват событий, генерируемых диалоговыми формами. Встрой – перехват событий, генерируемых диалоговыми формами, в прикладной программе.

Объекты ADO.NET

Текущая версия FastReport Studio предоставляет классы-обёртки для доступа к ADO.NET. Следующая таблица демонстрирует взаимосвязь между некоторыми объектами ADO.NET и их обертками.

.NET class	Класс обертка
DataTable	FrxDataTable
DataView	FrxDataView
DataSet	FrxDataSet

Классы обёртки поставляются в C# примере DataSetDemo, который находится в директории:

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

Эти классы-обёртки предоставляют прозрачный доступ к данным ADO.NET из движка FastReport. В основе, они используют объект TfrxUserDataSet для передачи данных от ADO.NET структур данных движку FastReport.
Нижеприведённый фрагмент кода демонстрирует, как использовать классы-обертки в 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 demonstration report from file
report.LoadReportFromFile("some_report.fr3");
// Following 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();
```

Пожалуйста, обратите внимание, что мы не можем наследовать объект FrxDataTable непосредственно от уже существующего объекта DataTable. Это означает, что если вы создали или получили экземпляр объекта DataTable, то Вы не имеете возможность привязать его к отчёту или конвертировать в объект FrxDataTable.

При использовании классов-обёрток следует обратить внимание на жизненный цикл объектов. Поскольку в managed средах объект фактически удаляется сборщиком мусора, а не после потери видимости или присвоении ему пустого значения, то возможно пересечение имён объектов в namespace FastReport. Такое пересечение может привести к нежелательным результатам поведения генератора отчётов.

Следует придерживаться основного правила: **Используйте только уникальные имена для именования каждого объекта доступа к данным в пределах приложения.**

Для использования отношения Master/Detail между ADO.NET таблицами, может быть использован следующий алгоритм:

1. Каждая запись главной таблицы должна иметь уникальный ключ для таблиц детализации.
2. Приложение обязано обрабатывать события OnFirst, OnNext, OnPrior и OnLast главной таблицы, чей тип может быть FrxDataTable или FrxDataView.
3. Таблицы детализации обязаны иметь тип FrxDataView.
4. Каждое событие навигации в главной таблице обязано устанавливать новый фильтр на таблицы детализации в соответствии с критериями, зависящими от значения полей главной таблицы

Следующий код демонстрирует, как установить отношение Maser/Detail между двумя объектами ADO.NET. В следующем примере основная таблица реализована как FrxDataTable и таблица детализации реализована как FrxDataView. Оба объекта связаны через поле «id».

```
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 );
}
```

Код регистрации события:

```
    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);
```

Обратите внимание: Классы-обёртки для ADO.NET не являются полным и законченным решением, они предоставляются в исходном коде для примера. Вы можете адаптировать эти классы для своих нужд

Вызов внешних функций

FastReport предоставляет возможность вызова внешних функций, определенных в прикладной программе. Как вы знаете, FastReport поставляется вместе со встроенным скриптовым интерпретатором, называемым FastScript. Но иногда возникает необходимость некоторую функциональность в прикладной программе. Причинами могут служить ускорение некоторой участка кода отчёта или сокрытие реализации некоторого алгоритма. Вызвать внешнюю функцию из FastReport очень просто – для этого необходимо выполнить три шага:

1. Зарегистрировать внешнюю функцию при помощи метода AddFunction объекта TfrxReport.
2. Добавить и зарегистрировать обработчик события OnUserFunction().
3. В обработчике события проверить имя и вызываемой функции и передать ей управление.

Зарегистрировать функцию очень просто:

```
report.AddFunction(
      "function SpellValue(Value: String): String;", // Определение функции
      "User functions",                               // Категория функции
      "The value spelled out function");              // Описание функции
```

Реализация обработчик события несколько отличается на различных платформах. Если Вы используете FastReport Studio в non-managed C++ приложениях, то Вам потребуется класс-обёртка для интерфейса IfxReportEvents. Мы настоятельно рекомендуем использовать ATL шаблон IDispatchImpl для этих целей. (смотрите C++ пример Callbacks demo). В случае managed кода, мы рекомендуем использовать делегаты. Заглянем в реализацию события OnUserFunction(). Формат данного события следующий.

```
HRESULT OnUserFunction(
```

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

Параметр MethodName является именем вызванной функции. Обработчик события может реализовать обработку нескольких событий, различая их по параметру MethodName.

Параметр Params содержит аргументы вызываемой функции. Параметры передаются в виде объекта SafeArray вариант. В данном случае первый элемент массива содержит первый аргумент функции, второй элемент массива – второй аргумент и т.д. Среда .NET предоставляет объект System.Array, который очень удобен для быстрого и удобного доступа к параметрам вызываемой функции. Параметр ResultValue принимает возвращаемое значение функции.

Следующий фрагмент кода на C# показывает пример реализации обработчика события OnUserFunction():

```
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;
    }
}
```

Полный пример реализации пользовательской функции на языке C# смотрите в директории:

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

Загрузка текста и картинок в отчёт из прикладной программы

Иногда может потребоваться задать текст или картину на странице отчёта из прикладной программы. Для этого существует только один способ – использовать событие OnBeforePrint. Это событие генерируется для каждого объекта перед его печатью на странице отчёта. Это событие имеет один параметр – Sender, имеющий тип IfrxReportComponent. Поскольку все объекты, которые могут быть отрисованы на странице отчета, наследованы от этого интерфейса, то прикладная программа имеет возможность отследить, какой объект начинает отрисовываться, путём проверки его имени. Затем, на основе имени объекта (или другого свойства объекта), прикладная программа имеет возможность изменить свойства этого объекта. В следующем примере показано как изменить текст у текстовой мемки и как задать картинку объекту TfrxPictureView:

```
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();
    }
}
```

В противовес событию OnBeforePrint существует событие OnAfterPrint(). Это событие должно быть использовано для освобождения любых ресурсов, выделенных в обработчике события OnBeforePrint():

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

Доступ к объектам отчёта из прикладной программы

Иногда бывает необходим доступ к объектам отчёта, таким как бэнды, мемки и т.д., из кода прикладной программы. В этом случае Вы можете запросить интерфейс на экземпляр любого объекта, используя имя объекта. FastReport предоставляет две формы метода FindObject для получению доступа к объекту отчёта. Первая форма метода FindObject принадлежит объекту TfrxReport. Этот метод позволяет запросить любой объект, содержащийся в отчёте, по его имени. Следующая форма метода FindObject – метод FindObject интерфейса IfrxComponent. Следует отметить, что все объекты отчёта, как визуальные, так и управляющие, поддерживают интерфейс IfrxReportComponent. Метод FindObject этого интерфейса позволяет запросить интерфейс IfrxReportComponent на экземпляр любого вложенного объекта. Поиск объекта также происходит по его имени. Далее показаны примеры на языках C++ и C# доступа к объектам отчета через интерфейс IfrxComponent .

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";
```

Следующий пример на языке C# показывает использование метода FindObject объекта TfrxReport. Например, для поиска мемки с именем «Memo3» используйте следующий код:

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

Создание шаблона отчёта из прикладной программы

FastReport предоставляет возможность создание отчёта из кода прикладной программы. Это означает, что Вы имеете возможность создавать шаблоны отчёта в момент исполнения прикладной программы и не сохранять эти шаблоны отчётов на диске. К примеру, Вы можете создавать отчёты, чей вид и расположение элементов основаны на структуре данных, с которыми происходит работа. Типы объектов отчёта, которые могут быть созданы «на лету», перечислены в следующей таблице:

Object name	Description
IfrxReportPage	Страница отчёта
IfrxReportTitle	Заголовок отчёта
IfrxMemoView	Текстовая мемка
IfrxReportSummary	Итоговый бэнд отчёта
IfrxDataBand	Бэнд данных
IfrxPictureView	Мемка, содержащая картинки
IfrxShapeView	Фигурные мемки
IfrxChartView	Диаграммы. Этот объект недоступен в свободной версии
IfrxSubreport	Вложенный отчёт
IfrxHeader	Заголовочный бэнд
IfrxFooter	Подвальный бэнд
IfrxMasterData	Бэнд с главными данными
IfrxDetailData	Детальный бэнд
IfrxSubdetailData	Субдетальный бэнд
IfrxDataBand4	Бэнд данных четвёртого уровня вложенности
IfrxDataBand5	Бэнд данных пятого уровня вложенности
IfrxDataBand6	Бэнд данных шестого уровня вложенности
IfrxPageHeader	Заголовок страницы
IfrxPageFooter	Подвал страницы
IfrxColumnHeader	Заголовок колонки
IfrxColumnFooter	Подвал колонки
IfrxGroupHeader	Заголовок группы
IfrxGroupFooter	Подвал группы
IfrxChild	Дочерний бэнд
IfrxOverlay	Перекрывающийся бэнд
IfrxCrossView	Объект CrossView
IfrxDBCrossView	Объект CrossView связанный с базой данных

Примеры динамического создания объектов расположены в следующих директориях (относительно пути установки FastReport Studio):

Язык	Путь к демонстрационной программе
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

При использовании VB6, подход к динамическому созданию объектов несколько отличается. Для этого должна быть использован метод **`CreateReportObjectEx`** вместо **`CreateReportObject`**:

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

Этот метод похож на метод CreateReportObject за исключением того, что параметр ObjectType представлен в виде строки, которая определяет тип создаваемого объекта. Существует простое правило преобразования имени интерфейса в тип объекта, передаваемый в функцию **`CreateReportObjectEx`**. Имя объекта образуется из названия его интерфейса посредством замены первого символа 'I' на символ 'T'. К примеру, IfrxMasterData становится TfrxMasterData, IfrxMemmoView становится TfrxMemoView и т.д.. Code below illustrates creatation report page.

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. Нижеприведённый код иллюстрирует создание страницы шаблона отчета:

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

```
page1 = report.CreateReportObjectEx( report, "TfrxReportPage", "ReportPage1")
```

Иерархия классов и интерфейсов

Этот раздел описывает иерархию объектов FastReport Studio. Цель данного раздела - показать прикладному программисту от чего наследуются различные объекты генератора отчётов. FR Studio спроектирована таким образом, что большинство интерфейсов предоставляют только специфичные для объекта методы и свойства. Для доступа к методам и свойствам, которые являются общими для нескольких объектов, необходимо запросить интерфейс класса предка. Поясним вышесказанное на примерах для различных языков программирования. Следующие примеры подразумевают что 'memo' – это объект, определённый интерфейсом IfrxMemoView.

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
```

Смотрите схему раздела «Иерархия объектов», которая построена в виде дерева наследования.

IfrxComponent

Интерфейс IfrxComponent является базовым интерфейсом для большинства объектов FastReport. Он содержит несколько методов и свойств, ответственных за расположение объекта на странице отчёта. Таким образом, для того чтобы установить размер и расположение объекта на странице, программист должен запросить этот интерфейс у объекта и установить необходимые значения. Помимо этого, IfrxComponent имеет свойство Name, которое определяет пользовательское имя объекта, например "Memo1", "Memo2",

"MasterData1" и т.д.

Обратите внимание, что методы CreateReportObject и FindObject класса TfrxReport, всегда возвращают интерфейс IfrxComponent на вновь созданный или найденный объект. Для того чтобы работать со специфическими свойствами и методами такого объекта, необходимо запросить соответственный интерфейс у этого объекта. Выше показано как это реализуется в различных языках программирования.

```
interface IfrxComponent : IDispatch {

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

Этот метод возвращает интерфейс IfrxComponent на дочерний объект, по его индексу. Индекс – целое число, которое может принимать значение от нуля до количества дочерних объектов минус единицы.

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

Это свойство возвращает базовое имя объекта. Т.е. имя класса объекта.

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

Это свойство возвращает описание объекта. Если объект не содержит описания, то возвращается пустая строка.

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

Это свойство возвращает количество дочерних объектов.

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

Это свойство управляет левой координатой объекта.

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

Это свойство управляет верхней координатой объекта.

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

Это свойство управляет шириной объекта

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

Это свойство управляет высотой объекта

```
    HRESULT _stdcall FindObject(
                  [in] BSTR ObjectName,
```

```
                [out, retval] IfrxComponent** Obj);
```

Этот метод производит поиск дочерних объекта по его имени. Если объект найдет, то значение obj принимает интерфейс IfrxComponent экземпляра найденного объекта. Обратите внимание, что данный метод позволяет находить только потомков объекта, для которого вызывается метод FindObject. Это означает, что Вы не сможете найти объект «Страница отчета», запросив её у объекта «Бэнд», поскольку бэнды располагаются на станице отчета, а не наоборот. Поэтому «Бэнд» является дочерним объектом по отношению к «Странице отчёта». Мы рекомендуем использовать для поиска метод FindObject объекта TfrxReport вместо этого метода.

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

Это свойство возвращает дополнительное имя объекта.

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

Это свойство возвращает пользовательское имя объекта.

```
};
```

Этот раздел описывает классы, которые унаследованы от IfrxComponent. (Точнее говоря, от класса TfrxComponent, который поддерживает интерфейс IfrxComponent)

IfrxReport

Этот интерфейс включает в себя свойства и методы объекта TfrxReport – основного объекта FastReport. Смотрите раздел «Обзор объектов FastReport» главу «TfrxReport» для детального описания объекта TfrxReport и поддерживаемых им интерфейсов. В том числе и IfrxReport. Примеры использования TfrxReport приводятся в разделе «использование объекта TfrxReport»

IfrxDataSet

Интерфейс IfrxDataSet является базовым интерфейсом для любого объекта, предоставляющего доступ к данным в FastReport. Прикладному программисту не обязательно использовать этот объект, поскольку всю функциональность предоставляют классы, порожденные от IfrxDataSet. Однако, этот интерфейс может быть использован для непосредственного доступа к данным из прикладной программы, используя для этого движок FastReport..

Замечание для .NET программистов: Обратите внимание на разницу в терминологии FastReport и ADO.NET. Говоря DataSet, программисты FastReport имеют в виду объект DataTable.NET. Так сложилось исторически, поскольку FastReport гораздо старше технологии .NET.

```
interface IfrxDataSet : IDispatch {
```

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

Это свойство управляет пользовательским именем датасета. Как было уже сказано в разделе «Использование объекта TfrxReport» избегайте создания объектов с одинаковым именем.

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

Это свойство управляет стартовой точкой навигации. Свойство может принимать следующие значения:
rbFirst – от начала данных
rbCurrent – от текущей записи.

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

Это свойство управляет числом элементов датасета. Оно имеет смысл, если свойство «RangeEnd» датасета установлено в значение reCount.

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

Это свойство управляет конечной точкой навигации. Свойство может принимать следующие значения:
reLast – до конца данных
reCurrent – до текущей записи
reCount – число записей определяется свойством датасета «RangeEndCount»

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

Это свойство возвращает число полей датасета.

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

Это свойство возвращает число записей датасета.

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

Этот метод возвращает значение поля, чьё имя задано параметром FieldName, активной записи.

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

Это свойство возвращает номер активной записи.

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

Этот метод переходит к первой записи датасета, то есть делает её активной (текущей).

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

Этот метод переходит к следующей записи датасета.

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

Этот метод переходит к предыдущей записи датасета.

```
};
```

IfrxUserDataSet

Этот интерфейс предоставляет свойства и методы объекта TfrxUserDataSet. Для подробного описания объекта TfrxUserDataSet и его интерфейсов смотрите раздел «Обзор объектов FastReport» главу «TfrxUserDataSet» Любой объект, поддерживающий интерфейс IfrxUserDataSet, также поддерживает интерфейс IfrxDataSet.

IfrxADOTable

Этот интерфейс предоставляет свойства и методы объекта TfrxADOTable. Смотрите раздел «Обзор объектов FastReport» главу «TfrxADOTable» для подробного описания объекта TfrxADOTable и его интерфейсов. Любой объект, поддерживающий интерфейс IfrxADOTable, также поддерживает интерфейс IfrxDataSet.

IfrxADOQuery

Этот интерфейс предоставляет свойства и методы объекта TfrxADOQuery. Смотрите раздел «Обзор объектов FastReport» главу «TfrxADOQuery» для подробного описания объекта TfrxADOQuery и его интерфейсов. Любой объект, поддерживающий интерфейс IfrxADOQuery, также поддерживает интерфейс IfrxDataSet.

TfrxADODatabase

Этот интерфейс предоставляет свойства и методы объекта TfrxADODatabase. Смотрите раздел «Обзор объектов FastReport» главу «TfrxADODatabase» для подробного описания объекта TfrxADODatabase и его интерфейсов.

IfrxPage

Этот интерфейс является базовым для объектов, описывающих страницы, например для TfrxReportPage. Он предоставляет свойство, которое отвечает за видимость страницы.

```
interface IfrxPage : IUnknown {

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

Получить текущее состояние видимости страницы

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

Установить видимость страницы.

```
};
```

IfrxReportPage

Интерфейс IfrxReportPage служит для управления объектом TfrxReportPage, который представляет собой страницу шаблона отчёта.

Обратите внимание, что одиночная страница шаблона отчёта может сгенерировать множество страниц подготовленного отчета, поскольку бэнды могут содержать множество записей.

```
interface IfrxReportPage : IDispatch {

    HRESULT _stdcall SetDefaults();
```

Этот метод сбрасывает свойства страницы в значения по умолчанию.

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

Управляет лотком принтера при печати первой страницы.

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

Управляет лотком принтера при печати последующих страниц. Последующие страницы могут возникнуть, если бэнд данных содержит множество записей, в этом случае одна страница шаблона отчета сгенерирует множество подготовленных страниц.

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

Это свойство управляет положением нижней границы отчёта на странице.

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

Это свойство управляет количеством столбцов на странице.

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

Это свойство управляет шириной столбцов.

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

Это свойство управляет положением столбцов. Свойство представляет собой строку,

разбитую на поля. Поля отделяются при помощи последовательности символов CR/LF. Каждое поле представляет собой текст, описывающий число с плавающей точкой.

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

Это свойство управляет привязкой датасета к странице.

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

Это свойство управляет режимом печати. Значение свойства может принимать одно из следующих значений, показанных в таблице:

dmNone	Дуплексный режим не используется
dmVertical	Вертикальный дуплексный режим
dmHorizontal	Горизонтальный дуплексный режим
dmSimplex	Вертикальный и горизонтальный дуплексный режим

Обратите внимание, что это свойство имеет смысл только для принтеров, которые поддерживают эти режимы.

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

Это свойство управляет горизонтальными направляющими. Свойство представляет собой строку, разбитую на поля. Поля отделяются при помощи последовательности символов CR/LF. Каждое поле представляет собой текст, описывающий число с плавающей точкой.

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

Это свойство позволяет изменять масштаб страницы в режиме дизайнера, для ручной правки бэндов, которые выходят за пределы страницы.

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

Это свойство управляет положением левой границы отчета на странице.

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

Это свойство управляет зеркальным расположением границ отчёта.

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

Это свойство управляет ориентаций бумаги, т.е. портретной или ландшафтной.

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

Это свойство управляет текстом структуры документа.

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

Это свойство запрещает или разрешает печать пустых страниц.

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

Это свойство позволяет экономить бумагу. К примеру, если две страницы имеют бэнды, которое занимают только половину листа, то установка этого свойства в значение «Истина» позволит напечатать оба бэнда на одной странице.

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

Это свойство управляет положением правой границы отчета на странице.

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

Это свойство управляет объектом вложенного отчёта, который встраивается в текущую страницу отчёта. Смотрите описание интерфейса «IfrxSubreport» для получения дополнительной информации.

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

По умолчанию, титул бэнда печатается после заголовка бэнда. Это свойство меняет порядок печати титула или заголовка бэнда.

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

Это свойство управляет положением верхней границы отчета на странице.

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

Это свойство управляет вертикальными направляющими. Свойство представляет собой строку, разбитую на поля. Поля отделяются при помощи последовательности символов CR/LF. Каждое поле представляет собой текст, описывающий число с плавающей точкой.

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

Это свойство управляет фоновой картинкой страниц. Параметр Value принимает значение хендла картинки.

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

Это свойство управляет шириной бумаги.

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

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

Это свойство управляет высотой бумаги.

};

IfrxView

Интерфейс IfrxView служит для управления классом TfrxView, который является предком для многих объектов отчёта, печатающихся на странице отчёта или бэнде. Этот интерфейс предоставляет доступ к свойствам, которые являются общими для всех потомков.

```
interface IfrxView : IDispatch {

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

Это свойство управляет именем поля датасета, назначенному объекту TfrxView.

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

Это свойство управляет пользовательским тегом. Этот тег удобен при использовании интерактивных отчетов. Например, поле тега может быть заполнено при помощи FastScript или в обработчике события OnBeforerPrint уникальным значением (ключом). Этот ключ впоследствии может быть использован в обработчике события OnClickObject. Вы можете использовать это поле для любых других целей.

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

Это свойство управляет строкой URL. Если это свойство задано и пользователь щёлкнет мышкой в окне предварительного просмотра на объект, то URL откроется в окне WEB браузера, выбранного в системе по умолчанию.

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

Это свойство управляет привязкой датасета к объекту TfrxView, по имени датасета.

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

Это свойство возвращает имя объекта TfrxView

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

Это свойство только для чтения, которое возвращает интерфейс на атрибуты рамки объекта. Для более подробной информации смотрите описание интерфейса IfrxFrame в текущем разделе. Подраздел «Вспомогательные интерфейсы»

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

Это свойство управляет поведением сдвига объекта. Параметр Value может принимать следующие значения:

sm_DontShift	Не сдвигать объект
sm_ShiftAlways	Сдвигать объект всегда
sm_WhenOverlapped	Сдвигать объект, если он перекрыт другим объектом

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

Это свойство определяет выравнивание объекта относительно бэнда или страницы. Параметр Value может принимать одно из следующих значений:

ba_None	Не выравнивать объект
ba_Left	Выравнивать по левому краю
ba_Right	Выравнивать по правому краю
ba_Center	Выравнивать по центру родительского объекта
ba_Width	Сделать ширину объекта равной ширине родительского объекта
ba_Bottom	Выровнять по низу родительского объекта
ba_Client	Привести размер объекта к размеру родительского объекта

```
};
```

IfrxMemoView

Этот интерфейс описывает «мемо» объекты. Назначение MemoView – отображать текст, поля баз данных, выражения и т.д. Все объекты TfrxMemoView поддерживают также IfrxView интерфейс. Интерфейс IfrxMemoView может быть использован как аргумент для метода CreateObject класса TfrxReport, таким образом, MemoView может быть создан динамически из прикладной программы.

Заметьте, что «мемки», которые размещены на бэндах данных, отрисовываются так много раз, сколько записей содержит датасет, назначенный этому бэнду.

```
interface IfrxMemoView : IDispatch {

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

Это свойство управляет атрибутом автоматической установки ширины «мемки». Если этот атрибут установлен в значение «Истина», то ширина «мемки» автоматически изменяется таким образом, чтобы соответствовать ширине её текста.

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

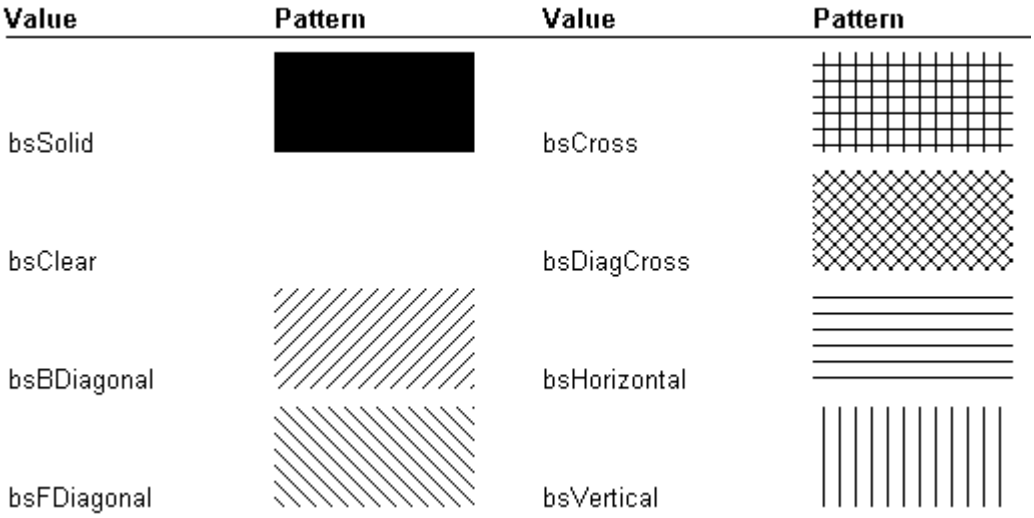
Это свойство определяет, могут ли текстовые объекты содержать выражения внутри текста. Значение по умолчанию – «Истина». Когда это свойство установлено в значение «Ложь», то ни одно выражение не будет вычислено в этом поле – отключены все объекты: переменные, выражение FastScript и любые другие преобразования данных. Таким образом, это поле будет показано без изменений.

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

Это свойство определяет, может ли текстовый объект содержать HTML теги.

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

Это свойство определяет тип используемой кисти. Доступные кисти показаны на картине ниже:




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

Это свойство определяет число точек (расстояние?) между символами.

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

Это свойство управляет обрезкой текста, выходящего за границы объекта.

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

Это свойство определяет цвет текста объекта. Параметр Value представлен в виде 32-х битного числа. Для более подробного описания цвета обращайтесь к документу "MSDN Platform SDK" тема «RGB».

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

Это свойство определяет поле, из которого объект будет получать данные.

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

Это свойство определяет датасет, из которого мемо будет получать данные.

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

Это свойство определяет имя датасета, к которому «привязана» мемка.

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

Этот метод возвращает интерфейс на объект TfrxDisplayFormat, который служит для управления преобразования вывода на экран. Например, при помощи TfrxDisplayFormat Вы можете изменять формат вывода числовых значений.

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

Это свойство определяет разделитель выражений в выражениях объекта TfrxMemoView.

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

Это свойство управляет тем, как будет растянут текст, который не помещается внутри мемки.

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

Это свойство возвращает интерфейс на объект TfrxFont, который управляет шрифтом мемки. Для описания интерфейса IfrxFont смотрите раздел "Иерархия объектов" подраздел "Вспомогательные объекты интерфейс IfrxFont"

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

Это свойство возвращает интерфейс на объект TfrxFrame, который управляет обрамлением мемки. Для описания интерфейса IfrxFrame смотрите раздел "Иерархия объектов" подраздел "Вспомогательные объекты интерфейс IfrxFrame"

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

Это свойство управляет левым выравниванием текста внутри мемки.

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

Это свойство управляет выравниванием текста по верху внутри мемки.

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

Это свойство управляет горизонтальным выравниванием текста внутри мемки. Возможные значения параметра Value перечислены в таблице ниже:

hAlignLeft	Включить выравнивание текста по левому краю
hAlignRight	Включить выравнивание текста по правому краю
hAlignCenter	Включить выравнивание текста по центру
hAlignBlock	Распределить текст равномерно по ширине

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

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

Это свойство управляет печатью нулей в мемке. Это свойство имеет значение только для числовых форматов мемки.

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

Это свойство возвращает интерфейс на объект TfrxGighlight, который управляет подсветкой мемок.

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

Это свойство управляет количеством точек (расстоянием?) между строками мемки.

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

Это свойство управляет текстом мемки. Текст мемки может содержать как обычный текст, так и выражения, включающие переменные, выражения FastScript и прочие выражения.

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

Это свойство управляет разрывом между параграфами текста.

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

Это свойство определяет, должен ли объект использовать шрифт родительского объекта. Если значение этого свойства установлено в «Истина», то мемка будет использовать шрифт родительского объекта, например бэнда или страницы отчёта.

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

Это свойство управляет вращением мемки. Параметр Value определяет величину угла в градусах, на которую следует повернуть объект. Вращение происходит в сторону часовой стрелки.

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

Это свойство управляет режимом чтения справа налево. Некоторые национальные языки используют такой вид письменности.

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

Это свойство определяет стиль мемки.

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

Это свойство управляет подавлением повторяющихся значений.

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

Это свойство управляет подчеркиванием текста мемки.

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

Это свойство управляет переносом слов. В настоящее время правила переноса работают только для русского языка.

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

Это свойство определяет, должны ли вставляться в текст мягкие переносы строк, для того чтобы текст не выходил из мемки по правой границе.

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

Это свойство управляет режимом вертикального выравнивания текста. Режимы вертикального выравнивания перечислены в таблице ниже:

vAlignTop	Включить выравнивание текста мемки по верхней границе
vAlignBottom	Включить выравнивание текста мемки по нижней границе
vAlignCenter	Включить выравнивание текста мемки по центру

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

Это свойство управляет режимом растягивания мемки. Режимы растягивания перечислены в таблице ниже:

sm_DontStretch	Не растягивать мемку – использовать оригинальный размер
sm_ActualHeight	Изменит высоту мемки таким образом, чтобы поместился весь текст.
sm_MaxHeight	Растянуть объект по высоте таким образом, чтобы он касался следующего

```
};
```

IfrxPictureView

Интерфейс IfrxPictureView служит для управления объектом TfrxPictureView, который отвечает раз картинки в отчёте. Все объекты TfrxPictureView поддерживают также IfrxView интерфейс. Интерфейс IfrxPictureView может быть использован при создании объекта при помощи метода CreateReportObject объекта TfrxReport, таким образом, объект TfrxPictureView может быть создан из прикладной программы.

```
interface IfrxPictureView : IDispatch {

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

Это свойство предоставляет доступ к картинке по её хэндлу. Вы можете динамически

менять картинки в отчёте в момент его генерации, установив обработчик события OnBeforePrint и подменяя хэндлы в этом обработчике.

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

Это свойство предоставляет доступ к картинке через поток данных. Вы можете динамически менять картинки в отчёте в момент его генерации, установив обработчик события OnBeforePrint, вызывая метод LoadViewFromStream в этом обработчике. Этот метод поддерживает COM и .NET потоки данных.

```
};
```

IfrxChartView

Этот интерфейс управляет встроенным объектом TfrxChartView, который реализован на основе библиотеки Tee Chart. Все объекты TfrxChartView поддерживают также IfrxView интерфейс. Интерфейс IfrxChartView может быть использован как аргумент для метода CreateObject класса TfrxReport, таким образом, ChartView может быть создан динамически из прикладной программы.

Обратите внимание, что объект IfrxChatrView не предоставляет всех возможностей по управлению объектом TfrxChartView. Большинством свойств объекта Вы можете управлять в дизайнере отчётов или при помощи FastScript.

```
interface IfrxChartView : IDispatch {

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

Этот метод используется для получения серии диаграммы по её номеру. Выходной параметр Value является интерфейсом объекта TfrxSerieaItem, который управляет серией диаграммы. Смотрите описание интерфейса IfrxSeriesItem ниже.

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

Этот метод добавляет новую серию к диаграмме и возвращает интерфейс к экземпляру вновь созданной серии. Параметр SeriesType может принимать одно из следующих значений.

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,

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

Это свойство возвращает число серий, которое зарегистрировано в объекте ChartView.

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

Это свойство управляет визуальным эффектом трехмерности диаграмм.

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

Это свойство управляет эффектом трехмерности границ диаграммы.

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

Эта два свойства управляют левой и нижней осями соответственно. Выходной параметр Value является интерфейсом IfrxChartAxis.

```
};
```

Серии диаграмм управляются при помощи интерфейса IfrxSeriesItem. Методы и свойства, предоставляемые интерфейсом IfrxSeriesItem общие для всех типов серий.

```
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);
```

Это свойство управляет датасетом, из которого серия получает данные.

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

Эти свойство определяют имя датасета, привязанного к серии данных.

```
    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);
```

Эти свойства управляют источником данных для осей X, Y, Z и других измерений соответственно.

```
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);
```

Эти свойства содержат данные для серий по осям X, Y, Z и других измерений соответственно.

```
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);
```

Это свойство управляет заголовками серий. Эти заголовки будут отображаться в окне легенды.

```
};

interface IfrxChartAxis: IDispatch {
```

Интерфейс IfrxChartAxis управляет осями диаграммы. К сожалению, документация FR Studio обновляется не так часто как исходный код, поэтому некоторые свойства и методы интерфейса IfrxChartAxis могут быть не описаны здесь.

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

Это свойство управляет автоматическим масштабированием осей. Значение по этого атрибута по умолчанию – «Истина». Перед установкой свойств минимального и максимального значений, установите это свойство Automatic в значение «Ложь». Использование минимального и максимального значений используется для ручной установки масштаба диаграммы.

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

Это свойство управляет минимальным значением данных диаграммы на текущей оси.

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

Это свойство управляет максимальным значением данных диаграммы на текущей оси.

```
};
```

IfrxShapeView

Интерфейс IfrxShapeView управляет объектом TfrxShapeView, который отвечает за отрисовку геометрических фигур в отчёте. Все объекты TfrxShapeView поддерживают также IfrxView интерфейс. Интерфейс IfrxShapeView может быть использован как аргумент для метода CreateObject класса TfrxReport, таким образом, ShapeView может быть создан динамически из прикладной программы.

```
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);
```

Это свойство управляет типом фигуры. Параметр Value может принимать одно из следующих значений, перечисленных в таблице ниже:

Параметр Value	Тип геометрической фигуры
skRectangle	Прямоугольник
skRoundRectangle	Прямоугольник с закруглёнными краями
skEllipse	Эллипс
skTriangle	Треугольник
skDiamond	Ромб
skDiagonal1	Диагональ 1
skDiagonal2	Диагональ 2

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

Это свойство возвращает интерфейс IfrxFrame, который служит для управления обрамлением объекта и его тенью.

```
};
```

IfrxOLEView

Интерфейс IfrxOleView управляет объектом TfrxOLEView, который служит для внедрения OLE объектов в отчёт.

Все объекты TfrxOLEView поддерживают также IfrxView интерфейс.

Интерфейс IfrxOLEView может быть использован как параметр метода CreateObject класса TfrxReport, таким образом, OLE View может быть создан динамически из прикладной программы.

```
interface IfrxOLEView : IDispatch {

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

Это свойство возвращает интерфейс IUnknown на контейнер внедрённого объекта. Смотрите описание OLE объектов в MSDN

```
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);
```

Это свойство управляет режимом растягивания объекта.

};

IfrxRichView

Интерфейс IfrxRichView управляет объектом TfrxRichView, который служит для отображения RTF документов.

Объекты TfrxRichView поддерживают интерфейс IfrxView.

Интерфейс IfrxRichView может быть использован как параметр метода CreateObject класса TfrxReport, таким образом, RichView может быть создан динамически из прикладной программы.

```
interface IfrxRichView : IDispatch {

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

Загружает RTF документ из потока данных. Поддерживаются COM и .NET потоки данных.

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

Сохраняет RTF документ в поток данных. Поддерживаются COM и .NET потоки данных.

};

IfrxSubreport

Интерфейс IfrxSubreport управляет объектом TfrxSubreport, который реализует вложенные отчёты.

Все объекты TfrxSubreport поддерживают интерфейс IfrxView.

Интерфейс IfrxSubreport может быть использован как параметр метода CreateObject класса TfrxReport, таким образом, вложенный отчёт может быть создан динамически из прикладной программы.

```
interface IfrxSubreport : IDispatch {

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

Это свойство определяет страницу вложенного отчёта, который будет отображать объект TfrxSubreport.

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

Это свойство управляет возможностью печати на родительском объекте.

```
};
```

IfrxCustomCrossView

Интерфейс IfrxCustomCrossView является базовым интерфейсом для интерфейсов IfrxCrossView и IfrxDBCrossView, которые управляют объектами TfrxCrossView и TfrxDBCrossView.

```
interface IfrxCustomCrossView : IDispatch {

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

Свойство управляет именами полей объекта IfrxCustomCrossView. Значения представлены в виде строки именами полей, разделёнными символами перевода строки (CR/LF). Если для поля задано несколько имён, то поле автоматически делится на вложенные поля. В случае, когда наследуемый от IfrxCustomView объект является DBCrossView, имя поля описывает имя моля набора данных (имя поля датасета).

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

Функции ячейки управляют операциями над ячейками для отображения итоговых значений для столбцов и строк. Параметр Index определяет номер строки или столбца, для которого задаётся функция. Тип функции определяется параметром Value. Возможные функции и их описания перечислены в следующей таблице:

cf_None	Не производить операции над ячейками
cf_Sum	Суммировать значения всех ячеек
cf_Min	Найти минимальное значение ячейки
cf_Max	Найти максимальное значение ячейки
cf_Avg	Вычислить среднее арифметическое ячеек
cf_Count	Посчитать количество ячеек

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

Это свойство предоставляет доступ к базовому интерфейсу IfrxCustomMemoView ячейки. Параметр Index используется для указания вложенных ячеек, если таковые присутствуют. Использование полученного интерфейса позволяет изменять свойства ячейки, такие как цвет фона, шрифт, рамку и другие.

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

Это свойство управляет именами столбцов. Значение представлено в виде строки, состоящей из названий столбцов, разделённых символом перевода строки (CR/LF).

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

Это свойство предоставляет доступ к базовому интерфейсу IfrxCustomMemoView заголовка столбца. Параметр Index используется для указания вложенных заголовков столбцов, если таковые присутствуют. Использование полученного интерфейса позволяет изменять свойства заголовка, такие как цвет фона, шрифт, рамку и другие.

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

Это свойство управляет правилом сортировки столбцов. Возможные правила сортировки и их описания перечислены в следующей таблице:

so_Ascending	Сортировать столбцы в порядке возрастания
so_Descending	Сортировать столбцы в порядке убывания
so_None	Не сортировать столбцы

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

Это свойство предоставляет доступ к базовому интерфейсу IfrxCustomMemoView ячейки общей суммы столбца. Параметр Index используется для указания вложенных ячеек общей суммы столбцов, если таковые присутствуют. Использование полученного интерфейса позволяет изменять свойства заголовка, такие как цвет фона, шрифт, рамку и другие.

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

Это свойство управляет именами строк. Значение представлено в виде строки, состоящей из названий строк, разделённых символом перевода строки (CR/LF).

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

Это свойство предоставляет доступ к базовому интерфейсу IfrxCustomMemoView заголовка строки. Параметр Index используется для указания вложенных заголовков строк, если таковые присутствуют. Использование полученного интерфейса позволяет изменять свойства заголовка, такие как цвет фона, шрифт, рамку и другие.

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

```
HRESULT _stdcall RowSort(
                [in] long Index,
                [in] frxCrossSortOrder Value);
```

Это свойство управляет правилом сортировки строк. Возможные правила сортировки и их описания перечислены в следующей таблице:

so_Ascending	Сортировать строки в порядке возрастания
so_Descending	Сортировать строки в порядке убывания
so_None	Не сортировать строки

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

Это свойство предоставляет доступ к базовому интерфейсу IfrxCustomMemoView ячейки общей суммы строки. Параметр Index используется для указания вложенных ячеек общей суммы строки, если таковые присутствуют. Использование полученного интерфейса позволяет изменять свойства заголовка, такие как цвет фона, шрифт, рамку и другие.

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

Это свойство определяет максимальную ширину ячейки.

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

Это свойство определяет минимальную ширину ячейки.

```
HRESULT _stdcall AddValues(
                [in] SAFEARRAY(VARIANT) Rows,
                [in] SAFEARRAY(VARIANT) Columns,
                [in] SAFEARRAY(VARIANT) Cells);
```

Этот метод используется для добавления данных в CrossView. Обратите внимание, что в случае использования DBCrossView, использование данного метода не рекомендуется.

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

Это свойство управляет выравниванием текста ячейки по левому краю.

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

Это свойство управляет выравниванием текста ячейки по верхнему краю.

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

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

Это свойство определяет, каким образом большие CrossTable будут разделяться на

страницы.

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

Это свойство определяет, нужно ли повторять заголовки строки и столбцов на новой странице, в случае если CrossView занимает более одной страницы.

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

Это свойство определяет, нужно ли печать заголовки столбцов.

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

Это свойство определяет, нужно ли печать ячейки общей суммы столбцов.

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

Это свойство определяет, нужно ли печать заголовки строк.

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

Это свойство определяет, нужно ли печать ячейки общей суммы строк.
};

IfrxCrossView

Интерфейс IfrxCrossView не предоставляет свойств и методов, а служит для создания объекта TfrxCrossView. Используйте этот интерфейс как аргумент функции CreateObject. Для работы с объектом CrossView, созданным функцией CreateObject или полученным из отчёта при помощи метода FindObject, используйте базовый интерфейс IfrxCustomCrossView. Объект CrossView служит для отображения табличной информации. Для добавления данных в CrossView используйте метод AddValues интерфейса IfrxCustomCrossView в обработчике события OnBeforePrint.

IfrxDBCrossView

Интерфейс IfrxDBCrossView не предоставляет свойств и методов, а служит для создания объекта TfrxDBCrossView. Используйте этот интерфейс как аргумент функции CreateObject. Для работы с объектом DBCrossView, созданным функцией CreateObject или полученным из отчёта при помощи метода FindObject, используйте базовый интерфейс IfrxCustomCrossView. Объект DBCrossView служит для отображения табличной информации полученной из базы данных. Ниже приведён фрагмент кода на языке C#, который динамически создает объект TfrxDBCrossView и устанавливает его свойства.

```
IfrxCustomCrossView ccv = report.CreateReportObject(
   page as IfrxComponent,
```

```
        typeof(IfrxDBCrossView).GUID,
        "DBCrossView") as IfrxCustomCrossView;

    ccv.ColumnFields = "Continent";
    ccv.RowFields = "Name";
    ccv.CellFields = "Population\r\nArea";
    ccv.MinWidth = 90;
    ccv.ShowColumnTotal = false;

    (ccv as IfrxView).DataSetName = "ADOTable1";
    (ccv.get_CellMemos(0) as IfrxMemoView).Color = 0xe0e0e0;
    (ccv.get_CellMemos(1) as IfrxMemoView).Color = 0xd0f0f0;
    (ccv.get_ColumnMemos(0) as IfrxMemoView).Color = 0x30e0e0;
    (ccv.get_ColumnMemos(0) as IfrxMemoView).Frame.Style =
frxFrameStyle.fs_Double;

    (ccv.get_RowMemos(0) as IfrxMemoView).Color = 0x30e0e0;
    (ccv.get_RowMemos(0) as IfrxMemoView).Frame.Style =
frxFrameStyle.fs_Double;
```

IfrxBand

Интерфейс IfrxBand служит для управления классом TfrxBand, который является предком любого бэнда. Этот интерфейс предоставляет доступ к свойствам и методам, которые являются общими для всех типов бэндов. Смотрите схему документа для просмотра списка всех описанных в документации бэндов.

```
interface IfrxBand : IDispatch {

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

Это свойство определяет, может ли бэнд быть разорван на другую страницу или обязан целиком поместиться на странице.

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

Это свойство определяет, должен ли бэнд печататься совместно с дочерним бэндом. Другими словами, это свойство позволяет переносить бэнд и дочерний бэнд на новый лист отчёта.

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

Это свойство управляет текстом схемы документа.

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

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

Это свойство управляет печатью каждой записи бэнда с новой страницы.

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

Это свойство управляет режимом растягивания бэнда.

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

Это свойство определяет должен ли печататься дочерний объект, если родительский объект невидим.

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

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

Это свойство определяет базовое имя объекта.
};

IfrxDataBand

Интерфейс IfrxDataBand служит для управления объектом TfrxDataBand, который является предком бэндов отображения данных. Объект обязан поддерживать интерфейс IfrxDataBand, если он поддерживает один из следующих интерфейсов: IfrxMasterData, IfrxDetailData, IfrxSubdetailData, IfrxDataBand4, IfrxDataBand5, IfrxDataBand6.

```
interface IfrxDataBand : IDispatch {

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

Это свойство управляет зазором между колонками бэнда.

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

Это свойство управляет шириной колонок бэнда.

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

Это свойство управляет количеством колонок бэнда.

```
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);
```

Это свойство позволяет назначить и получить датасет, привязанный к бэнду.

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

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

Это свойство определяет, должен ли печататься нижний колонтитул после каждой записи данных.

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

Это свойство определяет, должен ли бэнд печататься неразрывно с нижним колонтитулом.

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

Это свойство определяет, должен ли бэнд печататься неразрывно с верхним колонтитулом.

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

Это свойство определяет, должен ли бэнд печататься неразрывно с вложенными бэндами.

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

Это свойство определяет, должен ли бэнд данных печататься, если вложенный бэнд пустой.

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

Это свойство управляет количеством виртуальных записей в бэнде.

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

Этот метод сбрасывает привязку датасета к бэнду.

```
};
```

IfrxMasterData

Интерфейс IfrxMasterData случит для управления объектом TfrxMasterData. Объект TfrxMasterData – главный бэнд отчёта. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxDetailData

Интерфейс IfrxDetailData случит для управления объектом TfrxDetailData. Объект TfrxDetailData – используется для отображения вложенной (уточняющей) информации. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxSubdetailData

Интерфейс IfrxSubdetailData случит для управления объектом TfrxSubdetailData. Объект TfrxSubdetailData – используется для отображения вложенной (уточняющей) информации. Этот бэнд использует третий уровень вложенности, т.е. MasterBand -> DetailBand -> SubdetailBand. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxDataBand4

Интерфейс IfrxDataBand4 используется для управления бэндом четвёртого уровня вложенности. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxDataBand5

Интерфейс IfrxDataBand5 используется для управления бэндом пятого уровня вложенности. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxDataBand6

Интерфейс IfrxDataBand46используется для управления бэндом шестого уровня вложенности. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxReportTitle

Интерфейс IfrxReportTitle используется для управления объектом TfrxReportTitle – заголовком отчёта. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxReportSummary

Интерфейс IfrxReportSummary используется для управления объектом TfrxReportSummary – бэндом итоговой информации. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxHeader

Интерфейс IfrxHeader используется для управления объектом TfrxHeader – бэндом заголовка данных. Этот интерфейс может использоваться для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

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

Это свойство определяет, будет ли бэнд заголовка печататься на новой странице.

```
};
```

IfrxFooter

Интерфейс IfrxFooter используется для управления объектом TfrxFooter – бэндом нижнего колонтитула. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxPageHeader

Интерфейс IfrxPageHeader используется для управления объектом TfrxPageHeader – бэндом заголовка страницы. Этот интерфейс может использоваться для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

```
interface IfrxPageHeader : IDispatch {

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

Это свойство определяет, будет ли заголовок страницы печататься на первой странице.

```
};
```

IfrxPageFooter

Интерфейс IfrxPageFooter используется для управления объектом TfrxPageFoter – бэндом колонтитула страницы. Этот интерфейс может использоваться для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

```
interface IfrxPageFooter : IDispatch {

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

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

Это свойство определяет, будет ли колонтитул страницы печататься на первой странице.

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

Это свойство определяет, будет ли колонтитул страницы печататься на последней странице.

```
};
```

IfrxColumnHeader

Интерфейс IfrxColumnHeader используется для управления объектом TfrxColumnHeader – бэндом заголовка столбца. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxColumnFooter

Интерфейс IfrxColumnFooter используется для управления объектом TfrxColumnFooter – бэндом колонтитулом столбца. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxGroupHeader

Интерфейс IfrxGroupHeader используется для управления объектом TfrxGroupHeader – бэндом заголовком группировки. Заголовок группировки имеет свойства, с помощью которых можно задать правила группировки. Помимо этого, интерфейс IfrxGroupHeader может использоваться для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все базовые свойства бэнда.

```
interface IfrxGroupHeader : IDispatch {

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

Это свойство управляет правилом группировки. Пример правила группировки:
Copy(<Customers."Company">,1,1)+Copy(<Customers."City">,1,1).
Данное правило группирует записи в отчёте по первым буквам двух полей. Таким образом, сначала создаётся список значений по заданному выражению, затем группировка записей по вычисленному выражению.

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

Это свойство определяет, можно ли разрывать группу на разные страницы.

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

Это свойство определяет, нужно ли заголовок группы печатать на новой странице..

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

Это свойство содержит значение, по которому производится текущая группировка.
};

IfrxGroupFooter

Интерфейс IfrxGroupFooter используется для управления объектом TfrxGroupFooter – бэндом колонтитула группы. Интерфейс IfrxGroupFooter может использоваться для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все базовые свойства бэнда.

```
interface IfrxGroupFooter : IDispatch {

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

Это свойство скрывает нижний колонтитул группы, если была только одна строка мастера.

```
};
```

IfrxChild

Интерфейс IfrxChild используется для управления объектом TfrxChild – дочерним бэндом. Это бэнд используется для выравнивания данных в некоторых ситуациях. Пример использования дочернего бэнда освещён в «Руководстве пользователя FastReport». Интерфейс IfrxChild не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

IfrxOverlay

Интерфейс IfrxOverlay используется для управления объектом TfrxOverlay – накладного бэнда. Накладные бэнды могут использоваться для создания фонового текста или фоновых картинок. Этот интерфейс не предоставляет свойств и методов, а используется для динамического создания объекта методом CreateObject. Созданный объект поддерживает следующие интерфейсы: IfrxDataBand и IfrxBand, с помощью которых можно задать все необходимые свойства бэнда.

Вспомогательные интерфейсы

Эта глава описывает вспомогательные интерфейсы, которые введены для некоторых расширенных свойств объектов отчёта.

IfrxFont

Интерфейс IfrxFont служит для управления объектом TfrxFont, который используется для описания шрифтов элементов отчёта.

```
interface IfrxFont : IUnknown {

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

Это свойство отвечает за жирное начертание шрифта.

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

Это свойство отвечает за размер шрифта.

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

Это свойство описывает имя фонта.

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

Это свойство отвечает за наклонное начертание шрифта.

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

Это свойство отвечает за подчёркивание.

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

Это свойство позволяет управлять шрифтом через его хэндл. Для описания работы с хэндлом шрифта обращайтесь к MSDN.

```
    HRESULT _stdcall Charset([out, retval] frxCharset* Value);
    HRESULT _stdcall Charset([in] frxCharset Value);
```

Это свойство управляет кодировкой шрифта.

```
};
```

IfrxFrame

Интерфейс IfrxFrame служит для управления объектом TfrxFrame, который используется

для описания декораций элементов отчёта. К декорациям относятся Цвет рамки, Тень, Цвет тени, Ширина тени, Стиль, Тип рамки, Ширина рамки.

```
interface IfrxFrame : IUnknown {

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

Это свойство описывает цвет рамки.

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

Это свойство определяет, должен ли объект отбрасывать тень.

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

Это свойство управляет цветом тени.

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

Это свойство управляет шириной тени.

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

Это свойство определяет стиль рамки.

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

Это свойство определяет тип рамки.

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

Это свойство определяет ширину рамки.

```
};
```

IfrxHighlight

Интерфейс IfrxHighlight служит для управления объектом TfrxHighlight, который используется для подсветки элементов отчёта.

```
interface IfrxHighlight : IUnknown {

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

Это свойство определяет, активна ли подсветка.

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

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

Это свойство определяет цвет подсветки.

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

Это свойство определяет фонт подсветки.

```
};
```

Распространение приложений на базе FR Studio

Перед тем как Вы запустите Ваше приложение, использующее FR Studio, на другом компьютере, Вам потребуется переписать несколько файлов, которые перечислены в следующей таблице:

Путь установки по умолчанию.FR Studio:	`"C:\Program Files\FastReports\FastReport Studio\Bin\"`
Имя библиотеки:	`FasrReport3.dll`
Файлы языковых ресурсов	`Russian.frc, English.frc, и т.д.`

Помимо этого необходимо зарегистрировать динамическую библиотеку в операционной системе. Для этого необходимо вызвать внешнюю функцию динамической библиотеки DllRegisterServer из прикладной программы. Также Вы можете зарегистрировать динамическую библиотеку FastReport, используя командную строку. Для этого используйте следующую команду:

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

Для того чтобы снять регистрацию с библиотеки, используйте:

```
regsvr32.exe /u FastReport.dll
```

Данные команды необходимо выполнять, находясь в директории, содержащую файл FastReport.dll, или указывать полный путь к библиотеке.

Для установки национального языка пользовательского интерфейса компонентов FR Studio, внесите следящие данные в системный реестр:

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

Если Вы сомневаетесь, позволяет ли Ваша лицензия производить такие действия, обратитесь по адресу sales@fast-report.com.

Положения о двоичной совместимости между релизами

Мы не гарантируем двоичной совместимости между релизами. Вследствие динамичного развития продукта мы периодически изменяем интерфейсы (в большинстве случаев добавляем к ним новые свойства и методы). В результате, после апргрейда нашего продукта, прикладные программы, использующие FR Studio, могут начать работать неправильно, или даже приводить к Access Violation. Чтобы избежать такого поведения, вместе с апргейдом продукта необходимо перекомпилировать зависящие от него проекты.

В большинстве случае достаточно простой перекомпиляции, но иногда Вам понадобится внести изменения в Ваш код, чтобы он соответствовал изменениям новой версии продукта. Мы сожалеем о возможных неудобствах.