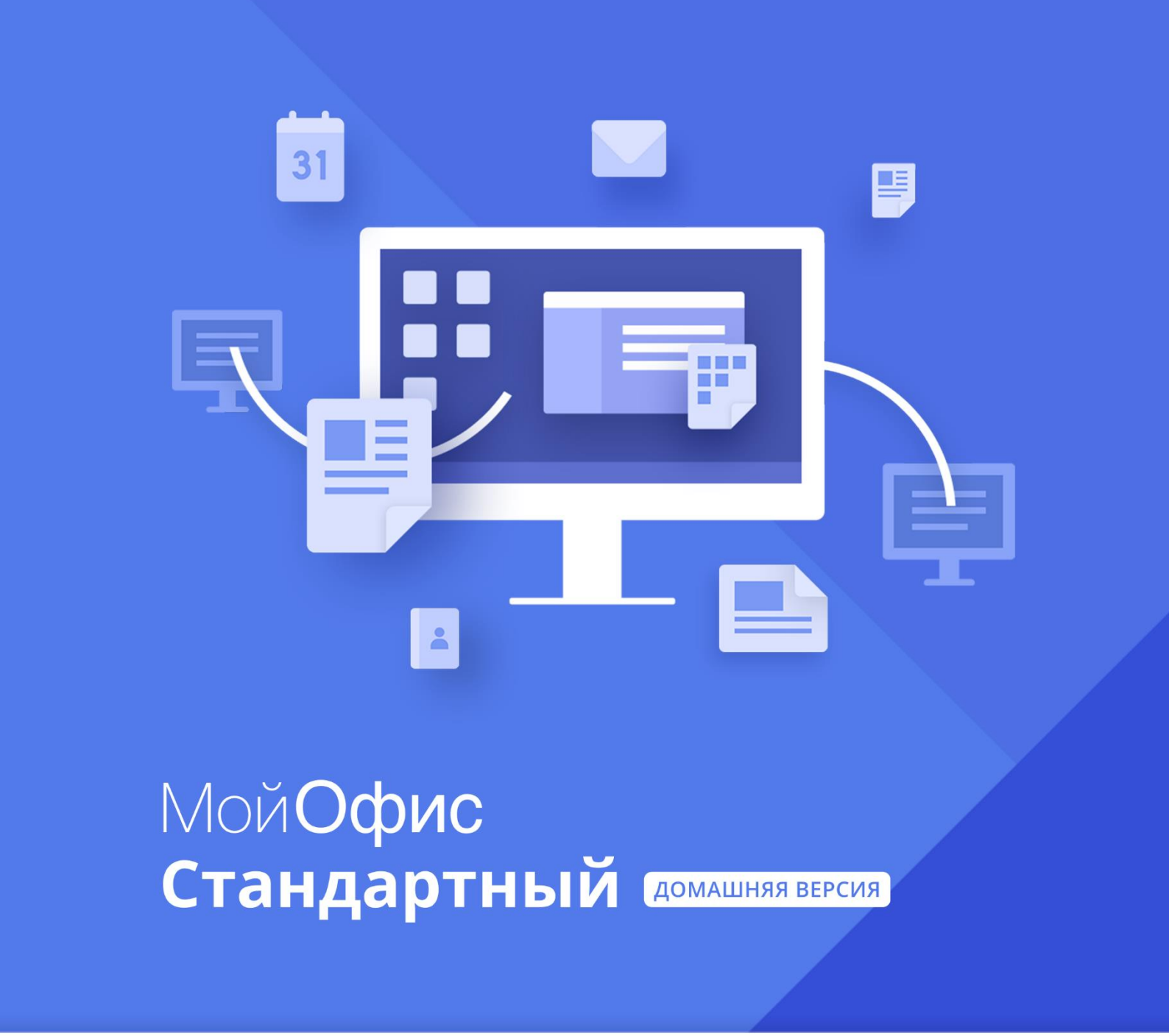


МойОфис
Стандартный
ДОМАШНЯЯ ВЕРСИЯ

Справочник макрокоманд

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС СТАНДАРТНЫЙ. ДОМАШНЯЯ ВЕРСИЯ»

СПРАВОЧНИК МАКРОКОМАНД НА ЯЗЫКЕ LUA

2020.03

На 92 листах

Москва
2021

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем. Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе используются следующие сокращения (см. Таблица 1):

Таблица 1 – Сокращения и расшифровки

Сокращение	Расшифровка
ОС	Операционная система
ПО МойОфис	Программное обеспечение «МойОфис Стандартный. Домашняя версия».
Объектная модель	Совокупность структур данных для управления содержимым текстового или табличного документа ПО МойОфис

1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначение программы

«МойОфис Стандартный. Домашняя версия» — продукт для работы с электронными документами специально для частных пользователей. С его помощью можно создавать и редактировать тексты и таблицы.

«МойОфис Текст» обеспечивает удобное и быстрое создание документов с использованием шаблонов, стилей и средств форматирования текста.

«МойОфис Таблица» – это приложение для быстрой и удобной работы с электронными таблицами и анализа данных.

Подробное описание функций ПО МойОфис приведено в документе «"МойОфис Стандартный. Домашняя версия". Функциональные возможности».

1.2 Макрокоманды ПО МойОфис

Макрокоманды ПО МойОфис представляют собой программы небольшого размера, с помощью которых автоматизируется выполнение продолжительных или часто встречающихся операций.

При работе с документом можно обнаружить, что одна и та же последовательность действий выполняется очень часто. Вместо того, чтобы каждый раз выполнять эту последовательность действий вручную, возможно создать макрокоманду для ее выполнения. Для разработки макрокоманд в ПО МойОфис используется язык программирования Lua.

Справочное руководство по языку программирования Lua на русском языке опубликовано по ссылке: http://lua.org.ru/contents_ru.html.

В языке программирования Lua таблицы – это единственная структура данных. Все структуры, которые предлагают другие языки программирования, в том числе массивы, объекты и другие, представлены в языке программирования Lua при помощи таблиц.

Структура данных «document», представляющая текущий открытый документ в ПО МойОфис, в терминах языка программирования Lua является таблицей со своим набором методов. Иные структуры данных для работы с отдельными ячейками, областями, диаграммами, свойствами текста и т. д. также являются таблицами с необходимым набором полей и методов.

Для управления содержимым документа используется объектная модель документа ПО МойОфис. В данном случае термин «объектная модель» обозначает всю совокупность структур данных для управления содержимым текстового или табличного документа ПО МойОфис.

Для управления текстовым или табличным документом ПО МойОфис используются одни и те же методы объектной модели. К примеру, объект Cell позволяет управлять как отдельной ячейкой электронной таблицы, так и ячейкой таблицы в текстовом документе.

Для работы с текстом макрокоманды используется редактор макрокоманд в составе текстового или табличного редактора ПО МойОфис. Редактор макрокоманд также предоставляет возможность исполнения макрокоманд и доступ к информации об ошибках их исполнения.

Текст макрокоманды сохраняется в текущий открытый документ ПО МойОфис. Макрокоманда в ПО МойОфис может быть сохранена только в документы формата XODT или ODT и XODS или ODS.

1.3 Перечень эксплуатационной документации

Настоящий документ содержит описание объектной модели документа ПО МойОфис и примеры ее использования, а также является справочником по возможностям объектной модели редакторов ПО МойОфис.

Вся необходимая информация по использованию макрокоманд в ПО МойОфис приведена в настоящем документе.

2 ОПИСАНИЕ ОПЕРАЦИЙ

2.1 Редактор макрокоманд

Для работы с макрокомандами используется редактор макрокоманд. Чтобы открыть окно редактора, в приложении «МойОфис Текст» или «МойОфис Таблица» выберите пункт командного меню **Инструменты** > **Редактирование макроса**.

Окно редактора макрокоманд содержит (см. Рисунок 1):

1. перечень созданных в документе макрокоманд;
2. кнопки для создания + и удаления — макрокоманд;
3. область ввода текста макрокоманд;
4. результат выполнения макрокоманд;
5. кнопку **Выполнить**.

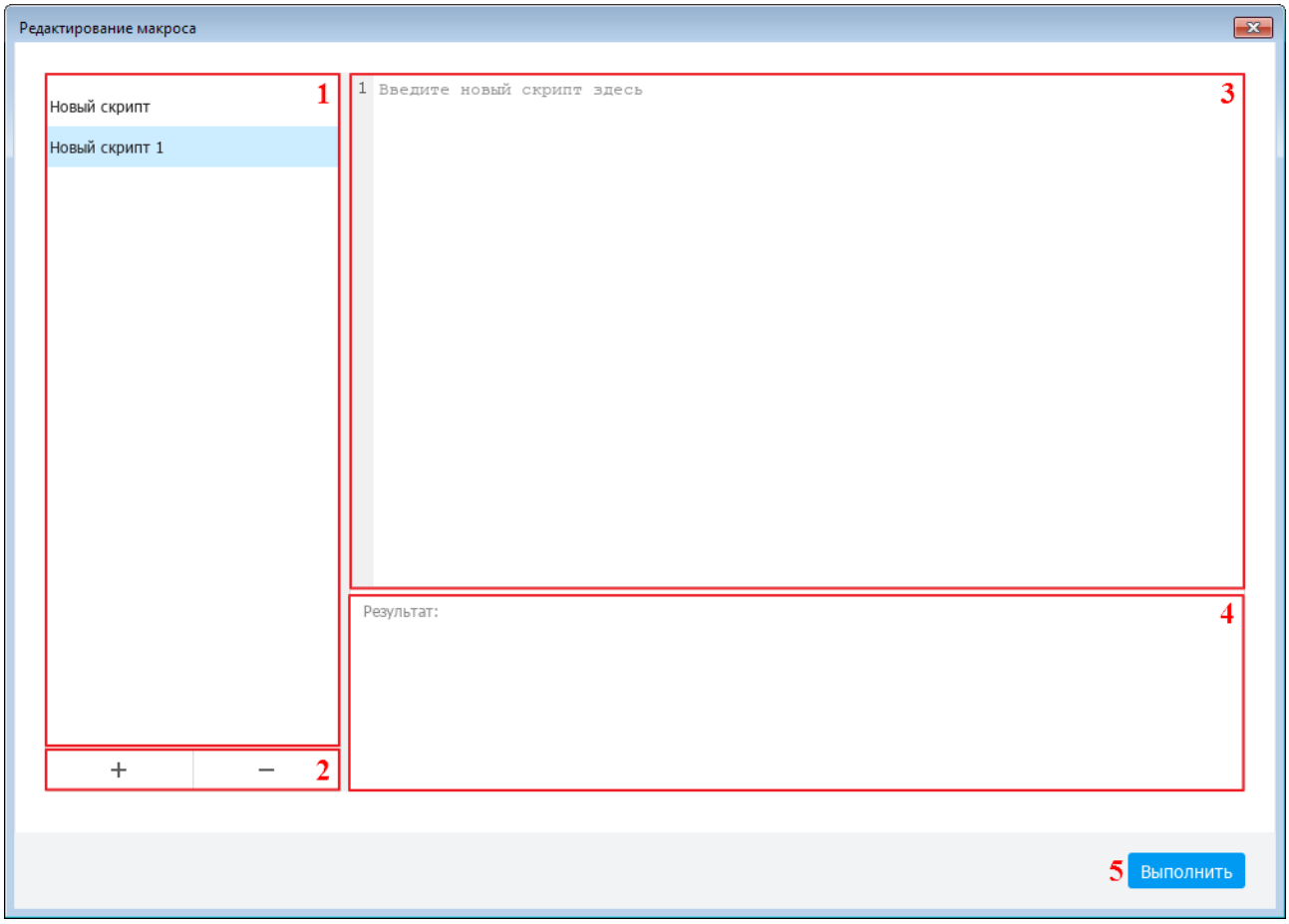


Рисунок 1 – Окно редактора макрокоманд

Для создания макрокоманды выполните следующие действия:

1. нажмите кнопку +;
2. введите наименование макрокоманды в соответствующей строке перечня макрокоманд. Чтобы сохранить название, нажмите клавишу **Enter** или щелкните мышью по любой области окна **Редактирование макроса**;
3. введите текст макрокоманды в области ввода.

Чтобы запустить выполнение макрокоманды, выберите ее в перечне макрокоманд и нажмите кнопку **Выполнить**.

Результат выполнения макрокоманды отображается в области, представленной под цифрой 4 (см. Рисунок 1).

Чтобы редактировать макрокоманду, выберите ее в перечне макрокоманд и внесите необходимые изменения в ее текст.

Чтобы удалить макрокоманду, выберите ее в перечне макрокоманд и нажмите кнопку −.

2.2 Пример подготовки и запуска макрокоманды

2.2.1 Редактирование и запуск макрокоманды в текстовом документе

Рассмотрим действия, необходимые для подготовки и запуска макрокоманды, на следующем примере: в текстовом редакторе требуется создать и запустить макрокоманду, которая выводит в первой строке текстового документа строку *"МОЙОФИС!"*.

Чтобы создать и запустить такую макрокоманду, необходимо выполнить следующие действия:

1. запустить приложение «МойОфис Текст» и открыть редактор макрокоманд. Для этого в командном меню выбрать пункт **Инструменты** > **Редактирование макроса**;
2. нажать кнопку + для создания новой макрокоманды. Указать имя макрокоманды. По умолчанию новой макрокоманде присваивается имя «Новый скрипт»;
3. ввести текст макрокоманды:

```
range = document:getRange();
startPos = range:getBegin();

textProp = range:getTextProperties();
textProp.italic = true;
textProp.allCapitals = true;

range:setTextProperties(textProp);
startPos:insertText("МойОфис!");
```

4. запустить макрокоманду нажатием на кнопку **Выполнить**. В случае успешного завершения работы макрокоманды редактор макрокоманд выведет сообщение «Макрос выполнен успешно»;
5. закрыть окно редактора макрокоманд, чтобы увидеть изменения в текстовом документе. В первой строке документа отобразится текст *"МОЙОФИС!"*;
6. сохранить текстовый документ. Для этого выбрать в командном меню пункт **Файл** > **Сохранить** / **Файл** > **Сохранить как** или нажать сочетание клавиш **Ctrl**+**S**.

   При сохранении необходимо выбрать тип файла «Text document (*.xodt)». В настоящий момент макрокоманда может быть сохранена только в формате XODT или ODT для текстовых документов и XODS или ODS для документов электронных таблиц.

2.2.2 Описание примера работы макрокоманды

Разберем работу макрокоманды, текст которой приведен в разделе 2.2.1.

С помощью последовательности вызовов устанавливается курсор в начало документа:

```
range = document:getRange();
startPos = range:getBegin();
```

Таблица **Document** (см. раздел 3.18) представляет текущий открытый текстовый документ.

Таблица **Range** (см. раздел 3.30) используется для того, чтобы предоставить доступ к любой части (фрагменту) содержимого документа.

В данном случае, переменная `range` содержит весь документ целиком. Вызов `range:getBegin()` устанавливает курсор в начало фрагмента, а в данном случае – в начало самого документа.

Следующая последовательность вызовов настраивает форматирование для документа:

```
textProp = range:getTextProperties();
textProp.italic = true;
textProp.allCapitals = true;
range:setTextProperties(textProp);
```

В результате выполнения `range:getTextProperties` переменной `textProp` присваивается экземпляр `TextProperties`, содержащий настройки форматирования текущего фрагмента документа.

Таблица **TextProperties** (см. раздел 3.29) позволяет управлять такими характеристиками как наименование и размер шрифта, цвет, начертание и т.п.

В данном примере устанавливаются две настройки форматирования:

- свойство `textProp.italic` принимает значение **true**, что равносильно нажатию кнопки *K* (**Курсив**) в пользовательском интерфейсе текстового редактора;
- свойство `textProp.allCapitals` принимает значение **true**, что равносильно нажатию кнопки AB (**Все прописные**) в пользовательском интерфейсе текстового редактора.

Следующий вызов `range:setTextProperties(textProp)` применяет новые настройки форматирования для документа. Теперь эти настройки форматирования будут применяться автоматически для вводимого текста.

Последний вызов вставляет в начало документа текст "МойОфис!":

```
startPos:insertText("МойОфис!");
```

При вставке текста автоматически применяются настройки форматирования, и итоговый текст отображается как "*МОЙОФИС!*" прописными буквами курсивом.

2.3 Преобразование макрокоманд на языке программирования VBA

Макрокоманды для пакета Microsoft Office, написанные на языке программирования VBA, не предназначены для исполнения в приложениях ПО МойОфис. Макрокоманды на языке программирования VBA предназначены для исполнения только в пакете Microsoft Office под управлением операционной системы семейства Microsoft Windows.

Однако большинство макрокоманд на языке программирования VBA возможно переписать на языке программирования Lua с использованием объектной модели ПО МойОфис.

ПО МойОфис является кроссплатформенным решением (решением не только для работы в операционных системах семейства Microsoft Windows), поэтому при переписывании макрокоманд с языка программирования VBA следует принимать во внимание следующие ограничения, связанные с операционными системами семейства Microsoft Windows:

- невозможность обращения к внешним приложениям с помощью технологий Component Object Model (COM);
- невозможность использования внешних динамических библиотек DLL.

Также в настоящее время в редакторе макрокоманд ПО МойОфис существует временное ограничение по работе в тексте макрокоманд с визуальными элементами, такими как выпадающие списки, переключатели и некоторыми другими.

3 СПРАВОЧНИК ТАБЛИЦ И МЕТОДОВ ДЛЯ РАБОТЫ С ОБЪЕКТНОЙ МОДЕЛЬЮ ПО МОЙОФИС

3.1 Форматы документов

В Таблица 2 приведены поддерживаемые форматы документов.

Таблица 2 – Форматы документов

Имя константы формата документа	Описание
`DocumentAPI.DocumentFormat_PlainText`	Используется для работы с файлами TXT
`DocumentAPI.DocumentFormat_DSV`	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем
`DocumentAPI.DocumentFormat_OXML`	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML
`DocumentAPI.DocumentFormat_ODF`	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010)
`DocumentAPI.DocumentFormat_HTML`	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в формате HTML средствами Document API в настоящий момент не поддерживается
`DocumentAPI.DocumentFormat_PDF`	Используется для работы с документами в формате Portable Dcoument Format (PDF), версии 1.4. Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b
`DocumentAPI.DocumentFormat_PDFA`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b). Средствами Document API поддерживается только операция экспорта документа в формат PDF/A-1b.

3.2 Типы документов

В Таблица 3 приведены поддерживаемые типы документов.

Таблица 3 – Типы документов

Имя константы типа документа	Описание
`DocumentAPI.DocumentType_Text`	Используется для работы с текстовыми документами – файлы DOCX, ODT, XODT, TXT
`DocumentAPI.DocumentType_Workbook`	Используется для работы с табличными документами – файлы XLSX, ODS, XODS
`DocumentAPI.DocumentType_Presentation`	Используется для работы с презентационными документами – файлы PPTX, ODP. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается.

3.3 Форматы экспорта документов

В Таблица 4 приведены поддерживаемые форматы экспорта документов.

Таблица 4 – Форматы экспорта документов

Имя константы формата экспорта документов	Описание
`DocumentAPI.ExportFormat_PDFA1`	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b)

3.4 Кодировки документов

В Таблица 5 приведены поддерживаемые кодировки документов.

Таблица 5 – Кодировки документов

Имя константы кодировки документа	Кодировка
`DocumentAPI.Encoding_Unknown`	Невозможно определить кодировку
`DocumentAPI.DocumentAPI.Encoding_UTF8`	UTF8
`DocumentAPI.Encoding_UTF16BE`	UTF16BE
`DocumentAPI.Encoding_UTF16LE`	UTF16LE
`DocumentAPI.Encoding_UTF32BE`	UTF32BE
`DocumentAPI.Encoding_UTF32LE`	UTF32LE
`DocumentAPI.Encoding_Windows1250`	Windows1250
`DocumentAPI.Encoding_Windows1251`	Windows1251
`DocumentAPI.Encoding_Windows1252`	Windows1252
`DocumentAPI.Encoding_ISO8859Part5`	ISO8859Part5
`DocumentAPI.Encoding_KOI8R`	KOI8R
`DocumentAPI.Encoding_KOI8U`	KOI8U
`DocumentAPI.Encoding_CP866`	CP866

3.5 Системы адресации ячеек

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в Таблица 6.

Таблица 6 – Системы адресации ячеек в табличном документе

Имя константы системы адресации ячеек	Система адресации ячеек	Описание
`DocumentAPI.FormulaType_A1`	A1	Вариант A1 соответствует наиболее распространенной системе адресации ячеек, при которой столбцы задаются буквами, а строки – числами
`DocumentAPI.FormulaType_R1C1`	R1C1	Вариант R1C1 соответствует альтернативной системе адресации ячеек, при которой столбцы и строки задаются числами

3.6 Типы линии

В Таблица 7 приведены типы линий.

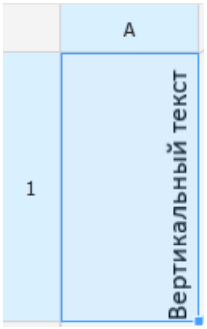
Таблица 7 – Типы линии

Имя константы типа линии	Описание
`DocumentAPI.LineStyle_NoLine`	Граница отсутствует
`DocumentAPI.LineStyle_Solid`	
`DocumentAPI.LineStyle_Dot`	
`DocumentAPI.LineStyle_Dash`	
`DocumentAPI.LineStyle_LongDash`	
`DocumentAPI.LineStyle_DashDot`	
`DocumentAPI.LineStyle_DotDotDash`	
`DocumentAPI.LineStyle_Double`	

3.7 Виды выравнивание текста, расположенного в ячейках вертикально

В Таблица 8 представлены виды выравнивания текста, расположенного в ячейках вертикально.

Таблица 8 – Виды выравнивание текста, расположенного в ячейках вертикально

Имя константы	Представление в интерфейсе
DocumentAPI.VerticalAlignment_Bottom	
DocumentAPI.VerticalAlignment_Center	
DocumentAPI.VerticalAlignment_Top	

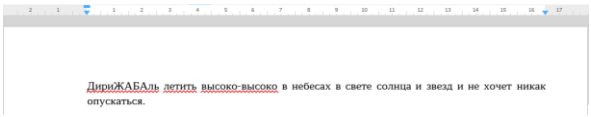
Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("A1")
local props = cell:getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cell:setCellProperties(props)
```

3.8 Виды выравнивание текста по горизонтали

В Таблица 9 представлены виды выравнивания по горизонтали текстовых фрагментов в текстовом редакторе или содержимого ячеек в табличном редакторе.

Таблица 9 – Виды выравнивания по горизонтали текстовых фрагментов

Имя константы	Описание
`DocumentAPI.Alignment_Default`	
`DocumentAPI.Alignment_Left`	по левому краю 
`DocumentAPI.Alignment_Center`	по центру 
`DocumentAPI.Alignment_Right`	по правому краю 
`DocumentAPI.Alignment_Justify`	по ширине 

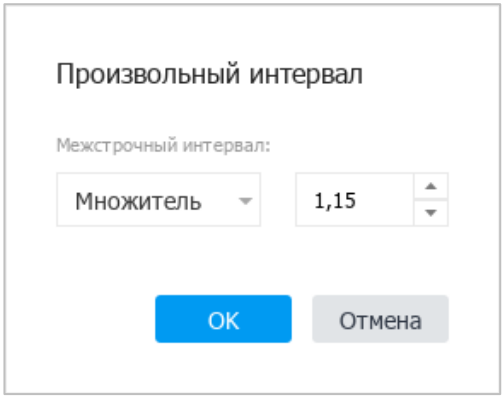
Пример:

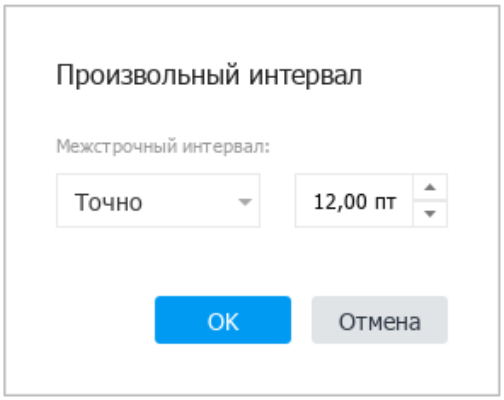
```
local para = document:getBlocks():getParagraph(0)
local props = para:getParagraphProperties()
props.alignment = DocumentAPI.Alignment_Center
para:setParagraphProperties(props)
```

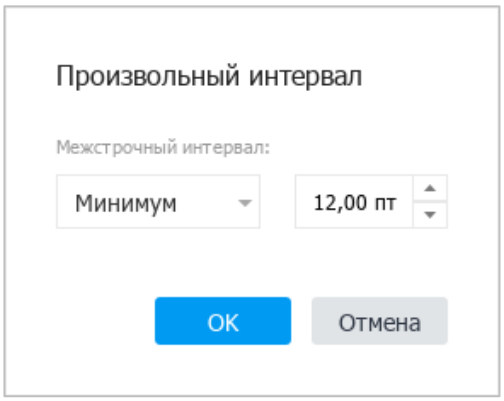

3.9 Виды межстрочного интервала

В Таблица 10 представлены виды межстрочного интервала для абзаца текста.

Таблица 10 – Виды межстрочного интервала

Имя константы	Описание
`DocumentAPI.LineSpacingRule_Multiple`	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: `pPr.lineSpacing = DocumentAPI.LineSpacing(1.15, DocumentAPI.LineSpacingRule_Multiple)` В данном примере используется значение множителя **1.15**. Действие команды аналогично настройке межстрочного интервала вручную с помощью окна **Произвольный интервал** (подробнее см. в документе «Руководство пользователя»). 

Имя константы	Описание
`DocumentAPI.LineSpacingRule_Exact`	Установка произвольного межстрочного интервала с использованием точного значения. При вызове необходимо указать точное значение, например: `pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_Exact)` В данном примере используется точное значение **12.0**. Действие команды аналогично настройке межстрочного интервала вручную с помощью окна **Произвольный интервал** (подробнее см. в документе «Руководство пользователя»). 

Имя константы	Описание
`DocumentAPI.LineSpacingRule_AtLeast`	Установка произвольного междустрочного интервала с использованием минимального значения. При вызове необходимо указать минимальное значение, например: `pPr.lineSpacing = DocumentAPI.LineSpacing(12.0, DocumentAPI.LineSpacingRule_AtLeast)` В данном примере используется точное значение **12.0**. Действие команды аналогично настройке межстрочного интервала вручную с помощью окна **Произвольный интервал** (подробнее см. в документе «Руководство пользователя»). 

Пример:

```
p = document:getBlocks():getParagraph(0)
pPr = p:getParagraphProperties()
pPr.lineSpacing = DocumentAPI.LineSpacing(5.0, DocumentAPI.LineSpacingRule_Multiple)
p:setParagraphProperties(pPr)
```


3.10 Виды отображения длинного текста

В Таблица 11 приведены виды отображения длинного текста в ячейках таблицы.

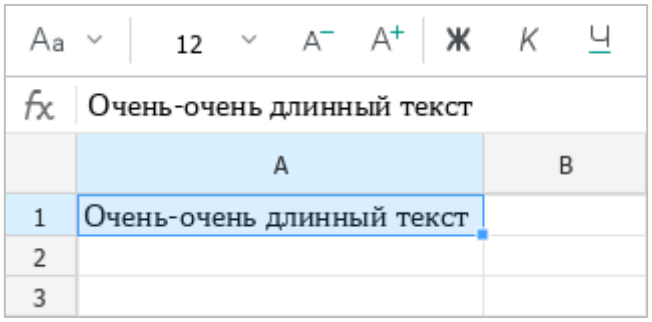
Таблица 11 – Виды отображения длинного текста

Имя константы	Описание
`DocumentAPI.TextLayout_SingleLine`	Отображение длинного текста в одну строку с наложением на соседние ячейки.
`DocumentAPI.TextLayout_WrapByWords`	Перенос данных внутри ячейки по словам.
`DocumentAPI.TextLayout_ShrinkSizeToFitWidth`	Текст или число отображаются так, чтобы содержимое поместилось в ячейке полностью. Установленный размер шрифта не изменяется.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell_A1 = tbl:getCell("A1")
local props = cell_A1:getCellProperties()
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
cell_A1:setCellProperties(props)
```

Результат:



3.11 Форматы ячеек таблицы

Поддерживаемые форматы представлены в таблице 12.

По умолчанию при создании документа всем ячейкам таблицы присваивается формат «Общий».

Таблица 12 – Поддерживаемые форматы ячеек таблицы

Имя константы	Пример	Описание
`DocumentAPI.CellFormat_General`	12	Формат ячейки «Общий». В этом формате в ячейке отображаются первые 9 символов числа, остальные доступны для просмотра в строке формул. Для дробных чисел в формате «Общий» незначащие нули в дробной части не отображаются. Числа, состоящие более чем из 12 символов, переводятся в экспоненциальную форму после завершения ввода в ячейку.
`DocumentAPI.CellFormat_Percentage`	120,00%	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
`DocumentAPI.CellFormat_Number`	12,00	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
`DocumentAPI.CellFormat_Text`	12	Формат ячейки «Текстовый».
`DocumentAPI.CellFormat_Currency`	1 200,00₽	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.

Имя константы	Пример	Описание
`DocumentAPI.CellFormat_Accounting`	1 200,00₽	Формат ячейки «Финансовый». Этот формат применяется для чисел, используемых в бухгалтерских документах. В формате «Финансовый» введенное число автоматически дополняется названием валюты, которая соответствует настройкам системы компьютера. Отрицательные числа в формате «Финансовый» заключаются в круглые скобки в ячейке, а в строке формул остаются в том виде, в котором они были введены.
`DocumentAPI.CellFormat_Date`	11.1.1900	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
`DocumentAPI.CellFormat_Time`	0:00:00	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
`DocumentAPI.CellFormat_Fraction`	12	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).
`DocumentAPI.CellFormat_Scientific`	1,2E+01	Формат ячейки «Экспоненциальный». Экспоненциальный (или научный) формат используется для представления больших чисел в короткой форме. Все введенные числа длиной более 12 символов автоматически переводятся в этот формат. В ячейке число в формате «Экспоненциальный» представлено следующим образом: – целая часть, всегда состоящая из одной цифры; – разделитель целой и дробной части; – дробная часть, по умолчанию состоящая из двух цифр; – показатель степени числа 10 в виде **E<знак показателя степени><показатель степени>**.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell_B1 = tbl:getCell("B1")
cell_B1:setFormat(DocumentAPI.CellFormat_General)
local cell_B2 = tbl:getCell("B2")
cell_B2:setFormat(DocumentAPI.CellFormat_Percentage)
local cell_B3 = tbl:getCell("B3")
cell_B3:setFormat(DocumentAPI.CellFormat_Number)
```

Результат:

	A	B
1	CellFormat.General	1
2	CellFormat.Percentage	100,00%
3	CellFormat.Number	1,00

3.12 Типы надстрочного и подстрочного форматирования

Типы представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе представлено в таблице 13.

Таблица 13 – Типы представления текста

Имя константы	Описание
`DocumentAPI.ScriptPosition_SuperScript`	Надстрочный знак (верхний индекс)
`DocumentAPI.ScriptPosition_SubScript`	Подстрочный знак (нижний индекс)
`DocumentAPI.ScriptPosition_NoramlScript`	Без указания индекса

3.13 Типы схем форматирования списков

Типы схем форматирования списков, которые могут быть применены к абзацам текста представлены в Таблица 14.

Таблица 14 – Типы схем форматирования списков

Имя константы типа форматирования списка	Описание
DocumentAPI.ListSchema_Unknown	Неизвестно
DocumentAPI.ListSchema_UnknownBullet	Список без маркера
DocumentAPI.ListSchema_UnknownNumbering	Нумерация без номера
DocumentAPI.ListSchema_BulletCircleSolid	Список с маркерами в виде круга
DocumentAPI.ListSchema_BulletCircleContour	Список с маркерами в виде окружности
DocumentAPI.ListSchema_BulletSquareSolid	Список с маркерами в виде квадрата
DocumentAPI.ListSchema_BulletDiamondDots	Список с маркерами в виде четырех ромбов
DocumentAPI.ListSchema_BulletHyphen	Список с маркерами в виде дефиса
DocumentAPI.ListSchema_BulletConcaveArrowSolid	Список с маркерами в виде вогнутой стрелки
DocumentAPI.ListSchema_BulletCheckmark	Список с маркерами в виде галочки
DocumentAPI.ListSchema_EnumeratorDecimalDot	Десятичная нумерация с точкой
DocumentAPI.ListSchema_EnumeratorDecimalDotMultiLevel	Многоуровневая десятичная нумерация с точкой
DocumentAPI.ListSchema_EnumeratorDecimalBracket	Десятичная нумерация со скобкой
DocumentAPI.ListSchema_EnumeratorLatinUppercaseDot	Нумерация латинскими прописными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseDot	Нумерация латинскими строчными буквами с точкой
DocumentAPI.ListSchema_EnumeratorLatinLowercaseBracket	Нумерация латинскими строчными буквами со скобкой

Имя константы типа форматирования списка	Описание
DocumentAPI.ListSchema_EnumeratorRomanUppercaseDot	Нумерация римскими прописными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorRomanLowercaseDot	Нумерация римскими строчными цифрами с точкой
DocumentAPI.ListSchema_EnumeratorDecimalRussianBracket	Десятичная нумерация через запятую со скобкой
DocumentAPI.ListSchema_EnumeratorRussianLowercaseBracket	Нумерация с русскими строчными буквами со скобкой

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```


3.14 Типы ориентации страницы

Типы ориентации страницы представлены в Таблица 15.

Таблица 15 – Типы ориентации страницы

Имя константы типа ориентации страницы	Описание	Изображение
`DocumentAPI.PageOrientation_Landscape`	Альбомная ориентация страницы	
`DocumentAPI.PageOrientation_Portrait`	Книжная ориентация страницы	

3.15 Типы колонтитулов

Типы колонтитулов представлены в Таблица 16.

Таблица 16 – Типы колонтитулов

Имя константы типа колонтитула	Описание
`DocumentAPI.HeaderFooterType_Header`	Верхний колонтитул
`DocumentAPI.HeaderFooterType_Footer`	Нижний колонтитул

3.16 Варианты кодов, возвращаемых после печати

В Таблица 17 представлены варианты кодов, возвращаемых после печати.

Таблица 17 – Варианты кодов, возвращаемых после печати

Код	Описание
`DocumentAPI.PrintDocumentResult_Success`	Печать прошла успешна
`DocumentAPI.PrintDocumentResult_CancelPrinting`	Печать была отменена
`DocumentAPI.PrintDocumentResult_NoPrinter`	Принтер не найден
`DocumentAPI.PrintDocumentResult_BlankDocument`	На печать отправлен пустой документ

3.17 Таблица DocumentAPI.Blocks

3.17.1 Метод Blocks:getParagraph

Возвращает абзац с указанным индексом. Нумерация индексов абзаца начинается с нуля.

Пример:

```
local para = document:getBlocks():getParagraph(0)
```

3.17.2 Метод Blocks:getTable

Для табличного документа возвращает лист (`worksheet`) с указанным номером. Нумерация листов начинается с нуля.

Для текстового документа возвращает таблицу с указанным порядковым номером. Нумерация таблиц начинается с нуля.

Пример:

```
local para = document:getBlocks():getTable(0)
```

3.17.3 Метод Blocks:enumerate

Возвращает таблицу объектов типа `Block`.

Пример:

```
for block in document:getBlocks():enumerate() do
    print(block:getRange():extractText())
end
```

3.17.4 Метод Blocks:enumerateTables

Возвращает таблицу объектов типа `Table`.

Пример:

```
for tbl in document:getBlocks():enumerateTables() do
    print(tbl:getName())
end
```

3.17.5 Метод Blocks:enumerateParagraphs

Возвращает таблицу объектов типа `Paragraph` (абзац).

Пример:

```
for para in document:getBlocks():enumerateParagraphs() do
    print(para:getRange():extractText())
end
```


3.18 Таблица DocumentAPI.Document

Таблица **DocumentAPI.Document** предоставляет доступ к содержимому открытого текстового или табличного документа.

Реализуется доступ к содержимому документа через таблицу `document`. К примеру,

```
local para = document:getBlocks():getParagraph(0)
```

предоставляет доступ к первому абзацу текстового документа.

3.18.1 Метод document:getBlocks

Метод предоставляет доступ к метатаблице **Blocks** (см. раздел 3.17) и далее к отдельным фрагментам (абзацам, таблицам и т. д.), из которых составлен документ.

Пример:

```
local blocks = document:getBlocks()
```

3.18.2 Метод document:getBookmarks

Метод предоставляет доступ к таблице закладок **Bookmarks** (см. раздел 3.33).

Пример:

```
local bmks = document:getBookmarks()
```

3.18.3 Метод document:getScripts

Метод предоставляет доступ к таблице макрокоманд **Scripts** (см. раздел 3.34), содержащихся в документе.

Пример:

```
local scripts = document:getScripts()
```

3.18.4 Метод document:getRange

Метод предоставляет доступ ко всему документу как области данных.

Пример:

```
local range = document:getRange()
```

3.18.5 Метод document:isChangesTrackingEnabled

Метод возвращает состояние включения отслеживания изменений в документе.

Пример:

```
local trackingChanges = "Disabled"
if document:isChangesTrackingEnabled() then
   trackingChanges = "Enabled"
end
```


3.18.6 Метод document:setChangesTrackingEnabled

Метод управляет состоянием включения отслеживания изменений в документе.

Пример:

```
if trackingChanges == "Disabled" then
  document:setChangesTrackingEnabled(true)
  if document:isChangesTrackingEnabled() then
     trackingChanges = "Enabled"
  end
end
```

3.18.7 Метод document:getComments

Метод обеспечивает доступ к комментариям, которые хранятся в документе.

Пример:

```
local comments = document:getRange():getComments()
```

3.18.8 Метод document:setPageProperties

Метод устанавливает ширину и высоту страниц в документе.

3.18.9 Метод document:setPageOrientation

Метод устанавливает альбомную, либо книжную ориентацию страниц в документе (см. раздел 3.14).

3.18.10 Метод document:enumerateSections

Возвращает таблицу объектов типа `Sections`.

3.19 Таблица DocumentAPI.Paragraph

Таблица **DocumentAPI.Paragraph** предоставляет доступ к свойствам абзаца.

3.19.1 Метод Paragraph:getParagraphProperties

Метод предоставляет доступ к таблице свойств форматирования абзаца **ParagraphProperties** (см. раздел 3.20), таким как выравнивание текста, межстрочные интервалы, отступы и т. п.

3.19.2 Метод Paragraph:setParagraphProperties

Метод предназначен для обновления таблицы свойств форматирования абзаца **ParagraphProperties** (см. раздел 3.20).

Пример:

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
para_props.alignment = DocumentAPI.Alignment_Right
para:setParagraphProperties(para_props)
```

3.19.3 Метод Paragraph:getListSchema

Метод возвращает схему форматирования абзаца (см. раздел 3.13) либо значение `nil`, если схема нумерации не установлена для абзаца.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
local schema = paragraph:getListSchema()
```

3.19.4 Метод Paragraph:setListSchema

Метод позволяет установить тип маркированного или нумерованного списка (см. раздел 3.13).

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

3.19.5 Метод Paragraph:getListLevel

Метод позволяет получить глубину вложенности элемента списка.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:getListLevel()
```


3.19.6 Метод Paragraph:setListLevel

Метод позволяет установить глубину вложенности элемента списка.

Значение может быть равным `nil`, если схема нумерации не установлена для абзаца. В этом случае будет установлено минимальное значение.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
local level = paragraph:setListLevel(1)
```

3.19.7 Метод Paragraph:increaseListLevel

Метод позволяет увеличить на единицу глубину вложенности элемента списка.

В случае, если максимальный уровень уже установлен, увеличения не происходит.

Пример:

```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:increaseListLevel()
```

3.19.8 Метод Paragraph:decreaseListLevel

Метод позволяет уменьшить на единицу глубину вложенности элемента списка.

В случае, если минимальный уровень уже установлен, уменьшения не происходит.

Пример:

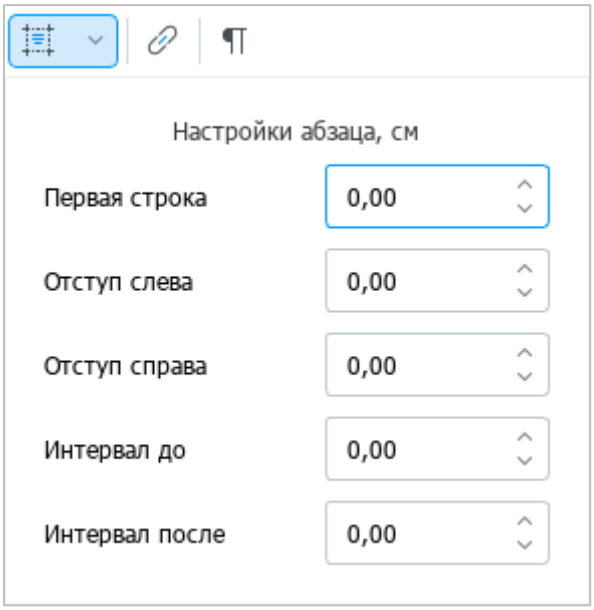
```
local paragraph = document:getBlocks():getParagraph(0)
paragraph:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraph:decreaseListLevel()
```


3.20 Таблица DocumentAPI.ParagraphProperties

Таблица **DocumentAPI.ParagraphProperties** предназначена для управления свойствами форматирования абзаца.

Описание полей таблицы **DocumentAPI.ParagraphProperties** представлено в таблице 18.

Таблица 18 – Описание полей таблицы DocumentAPI.ParagraphProperties

Поле	Свойства	Описание
`ParagraphProperties.afterSpacing`	[ReadWrite]	Установка величины расстояния после абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, в поле **Интервал после** (подробнее см. в документе «Руководство пользователя»). 
`ParagraphProperties.beforeSpacing`	[ReadWrite]	Установка величины расстояния до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца,** (см. рисунок выше), в поле **Интервал до** (подробнее см. в документе «Руководство пользователя»).
`ParagraphProperties.alignment`	[ReadWrite]	Выравнивание текстового фрагмента по горизонтали. Список допустимых значений см. в разделе 3.8.

Поле	Свойства	Описание
ParagraphProperties.firstLineIndent	[ReadWrite]	Расстояние от левого поля документа до первой строки в абзаце с учетом отступа слева. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Первая строка** (подробнее см. в документе «Руководство пользователя»).
ParagraphProperties.leftIndent	[ReadWrite]	Расстояние от левого поля документа до абзаца (отступ слева). При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ слева** (подробнее см. в документе «Руководство пользователя»).
ParagraphProperties.lineSpacing	[ReadWrite]	Расстояние между строк одного абзаца (межстрочный интервал). Для управления значением межстрочного интервала используйте значения, представленные в разделе 3.9.
ParagraphProperties.rightIndent	[ReadWrite]	Расстояние от правого поля документа до абзаца. При работе с пользовательским интерфейсом приложения соответствует значению, указанному в диалоговом окне **Настройки абзаца**, (см. рисунок выше), в поле **Отступ справа** (подробнее см. в документе «Руководство пользователя»).

Пример:

```
local para = document:getBlocks():getParagraph(0)
local para_props = para:getParagraphProperties()
--
para_props.afterSpacing = 28.3 -- значение соответствует 1 см
para_props.beforeSpacing = 28.3 -- значение соответствует 1 см
para_props.alignment = DocumentAPI.Alignment_Center
para_props.firstLineIndent = 28.3 -- значение соответствует 1 см
para_props.leftIndent = 28.3 -- значение соответствует 1см
para_props.lineSpacing = DocumentAPI.LineSpacing(5.0,
DocumentAPI.LineSpacingRule_Multiple)
para_props.rightIndent = 28.3 -- значение соответствует 1см
--
para:setParagraphProperties(para_props)
```


3.21 Таблица DocumentAPI.CellPosition

Таблица **DocumentAPI.CellPosition** позволяет задать координаты ячейки электронной таблицы или таблицы в составе текстового документа.

Нумерация строк и столбцов начинается с нуля, так, что позиция ячейки A1 имеет координаты (0, 0).

Также для указания адреса ячейки можно использовать строку вида "A1" в качестве параметра метода `getCell`.

Пример:

```
local table = document:getBlocks():getTable( 0 ) -- первый лист книги
local cell = table:getCell ( DocumentAPI.CellPosition ( 2, 0 ) ) -- ячейка A3
```

либо

```
local table = document:getBlocks():getTable( 0 ) -- первый лист книги
local cell = table:getCell ( "A3" ) -- ячейка A3
```

3.21.1 Метод CellPosition:toString

Возвращает координаты ячейки в формате `(row: R, column: C)`, где `R` и `C` - номер строки и столбца соответственно.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local pos = DocumentAPI.CellPosition(0,0)
print( pos:toString() ) - (row: 0, column: 0)
```

3.21.1.1 Поле CellPosition.row

Номер строки в позиции ячейки. Нумерация строк начинается с нуля.

3.21.1.2 Поле CellPosition.column

Номер столбца в значении ячейки. Нумерация столбцов начинается с нуля.

3.22 Таблица DocumentAPI.ColorRGBA

Таблица **DocumentAPI.ColorRGBA** предназначена для настройки цвета отображения текста и линий. Используется четырехканальный формат, содержащий данные для красного (`r`), голубого (`b`), зеленого (`g`) цветов и альфа-канала (`a`).

Описание таблицы **DocumentAPI.ColorRGBA** представлено в таблице 19.

Таблица 19 – Описание таблицы DocumentAPI.ColorRGBA

Цвет	Свойства	Описание
`r`	[ReadWrite]	Значение от 0 до 255 для установки интенсивности красного цвета.
`g`	[ReadWrite]	Значение от 0 до 255 для установки интенсивности зеленого цвета.
`b`	[ReadWrite]	Значение от 0 до 255 для установки интенсивности голубого цвета.
`a`	[ReadWrite]	Значение от 0 до 255 для регулировки прозрачности. Значение 255 соответствует полностью непрозрачному цвету.

Пример:

```
local line_prop = DocumentAPI.LineProperties()
line_prop.color = DocumentAPI.ColorRGBA(55, 146, 179, 200)
```


3.23 Таблица DocumentAPI.TextOrientation

Таблица **DocumentAPI.TextOrientation** предоставляет доступ к свойствам ориентации текста в ячейке, фигуре и т. д.

3.23.1 Метод TextOrientation:getAngle

Возвращает угол ориентации текста в ячейке. Значение угла указывается в градусах.

3.24 Таблица DocumentAPI.CellProperties

Таблица **DocumentAPI.CellProperties** предназначена для выравнивания содержимого в ячейках таблицы. Описание полей таблицы **DocumentAPI.CellProperties** представлено в таблице 20.

Таблица 20 – Описание полей таблицы DocumentAPI.CellProperties

Поле	Свойства	Значение
`CellProperties.verticalAlignment`	[ReadWrite]	Настройка вертикального выравнивания значения ячейки (см. раздел 3.7).
`CellProperties.backgroundColor`	[ReadWrite]	Настройка цвета фона ячейки. См. описание таблицы **DocumentAPI.ColorRGBA** в разделе 3.22.
`CellProperties.textLayout`	[ReadWrite]	Настройка вариантов отображения значения ячейки (см. раздел 3.10).
`CellProperties.textOrientation`	[ReadWrite]	Настройка свойств ориентации текста в ячейке. См. описание таблицы **DocumentAPI.TextOrientation** в разделе 3.23

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("D2") --(DocumentAPI.CellPosition(3,1))
local props = cell:getCellProperties()
--
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
props.textLayout = DocumentAPI.TextLayout_ShrinkSizeToFitWidth
props.backgroundColor = DocumentAPI.ColorRGBA(255,255,0,1)
--
cell:setCellProperties(props)
```


3.25 Таблица DocumentAPI.LineProperties

Таблица **DocumentAPI.LineProperties** предназначена для установки параметров линии, таких как тип линии, ее ширина или цвет.

3.25.1 Поле LineProperties.style

Поле предназначено для установки типа линии. Допустимые значения типа линии представлены в разделе 3.6.

3.25.2 Поле LineProperties.width

Поле предназначено для установки ширины линии.

3.25.3 Поле LineProperties.color

Поле предназначено для установки цвета линии.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")
--
line_prop = DocumentAPI.LineProperties()
line_prop.style = DocumentAPI.LineStyle_Solid
line_prop.width = 1.5
line_prop.color = DocumentAPI.ColorRGBA(55, 146, 179, 200)
--
rb = DocumentAPI.Borders()
rb = rb:setTop(line_prop)
local brds = cell:setBorders(rb)
```


3.26 Таблица DocumentAPI.Borders

Таблица **DocumentAPI.Borders** предназначена для оформления границ отдельной ячейки таблицы (см. Таблица 21). Параметры линии, такие как тип линии, ее ширина и цвет, задаются с помощью таблицы **DocumentAPI.LineProperties** (см. раздел 3.25).

Таблица 21 – Описание методов таблицы DocumentAPI.Borders

Метод	Описание
`Borders:setLeft`	Установка левой границы ячейки.
`Borders:setRight`	Установка правой границы ячейки.
`Borders:setTop`	Установка верхней границы ячейки.
`Borders:setBottom`	Установка нижней границы ячейки.
`Borders:setDiagonalDown`	Установка диагональной линии. 
`Borders:setDiagonalUp`	Установка диагональной линии. 
`Borders:setOuter`	Установка внешних границ ячейки.
`Borders:setDiagonals`	Установка обеих типов диагональных линий одновременно.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("C3")
--
line_prop = DocumentAPI.LineProperties()
line_prop.style = DocumentAPI.LineStyle_Dash
line_prop.width = 1.5
line_prop.color = DocumentAPI.ColorRGBA(55, 146, 179, 200)
--
rb = DocumentAPI.Borders()
rb = rb:setLeft(line_prop)
rb = rb:setRight(line_prop)
rb = rb:setTop(line_prop)
rb = rb:setBottom(line_prop)
--
local brds = cell:setBorders(rb)
```

Результат:

3.27 Таблица DocumentAPI.RangeBorders

Таблица **DocumentAPI.RangeBorders** предназначена для оформления границ области ячеек на листе таблицы.

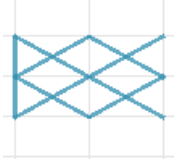
Таблица 22 – Описание методов таблицы DocumentAPI.RangeBorders

Метод	Описание
`RangeBorders:setLeft`	Установка левой границы ячейки.
`RangeBorders:setRight`	Установка правой границы ячейки.
`RangeBorders:setTop`	Установка верхней границы ячейки.
`RangeBorders:setBottom`	Установка нижней границы ячейки.
`RangeBorders:setDiagonalDown`	Установка диагональной линии. 
`RangeBorders:setDiagonalUp`	Установка диагональной линии. 
`RangeBorders:setOuter`	Установка внешних границ ячейки.
`RangeBorders:setDiagonals`	Установка обеих диагональных линий одновременно.

Пример:

```
line_prop = DocumentAPI.LineProperties()
line_prop.style = DocumentAPI.LineStyle_Solid
line_prop.width = 1.5
line_prop.color = DocumentAPI.ColorRGBA(55, 146, 179, 200)
--
local tbl = document:getBlocks():getTable(0)
local cell_rng = tbl:getCellRange("C3:D4")
--
local rb = DocumentAPI.RangeBorders()
rb = rb:setLeft(line_prop)
rb = rb:setRight(line_prop)
rb = rb:setTop(line_prop)
rb = rb:setBottom(line_prop)
rb = rb:setDiagonals(line_prop)
--
cell_rng:setBorders(rb)
```

Результат:



3.28 Таблица DocumentAPI.CellRange

Таблица **DocumentAPI.CellRange** предоставляет доступ к указанному диапазону ячеек таблицы.

3.28.1 Метод CellRange:enumerate

Метод возвращает коллекцию ячеек в диапазоне.

Пример:

```
-- Печать значений ячеек в диапазоне B3:C4
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
for cell in rng:enumerate() do
print(cell:getFormattedValue())
end
```

3.28.2 Метод CellRange:getBeginRow

Метод возвращает индекс строки первой ячейки диапазона. Нумерация строк начинается с нуля.

Пример:

```
local tbl = document:getBlocks( ):getTable( 0 )
local rng = tbl:getCellRange( "B3:C4" )
print( rng:getBeginRow( ) ) - 2
```

3.28.3 Метод CellRange:getBeginColumn

Метод возвращает индекс столбца первой ячейки диапазона. Нумерация столбцов начинается с нуля.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getBeginColumn()) - 1
```

3.28.4 Метод CellRange:getLastRow

Метод возвращает индекс строки последней ячейки диапазона. Нумерация строк начинается с нуля.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getLastRow()) -- 3
```


3.28.5 Метод CellRange:getLastColumn

Метод возвращает индекс столбца последней ячейки диапазона. Нумерация столбцов начинается с 0.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
print(rng:getLastColumn()) -- 2
```

3.28.6 Метод CellRange:setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов таблицы **RangeBorders** (см. раздел 3.27).

3.28.7 Метод CellRange:getCellProperties

Метод возвращает набор свойств форматирования для диапазона. Возвращаемая структура содержит свойства, общие для всех ячеек диапазона.

3.28.8 Метод CellRange:setCellProperties

Метод предназначен для установки свойств всех ячеек диапазона.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("B3:C4")
---
local props = DocumentAPI.CellProperties()
props.backgroundColor = DocumentAPI.ColorRGBA(55, 146, 179, 200)
rng:setCellProperties(props)
```

3.28.9 Метод CellRange:merge

Метод объединяет несколько ячеек таблицы в одну. Группа ячеек (диапазон) формируется с помощью таблицы **CellRange**. Содержимое крайней левой ячейки диапазона помещается в объединенной ячейке.

Пример:

```
-- Объединение ячеек A1 и A2 на первом листе табличного документа
local tbl = document:getBlocks():getTable(0)
tbl:getCellRange("A1:A2"):merge()
```


3.29 Таблица DocumentAPI.TextProperties

Таблица **DocumentAPI.TextProperties** предназначена для форматирования текста. Описание полей таблицы **DocumentAPI.TextProperties** представлено в таблице 23.

Таблица 23 – Описание полей таблицы DocumentAPI.TextProperties

Поле	Значение	Описание
`TextProperties.fontName`	Строковое, read/write	Наименование шрифта, использованного для оформления фрагмента документа.
`TextProperties.fontSize`	Числовое с плавающей точкой, read/write	Размер шрифта, использованного для оформления фрагмента документа.
`TextProperties.bold`	true/false	Значение **true** устанавливает жирное начертание для указанного фрагмента текста.
`TextProperties.italic`	true/false	Значение **true** устанавливает начертание курсивом для указанного фрагмента текста.
`TextProperties.underline`	true/false	Значение **true** устанавливает подчеркивание для указанного фрагмента текста.
`TextProperties.strikethrough`	true/false	Значение **true** устанавливает начертание «зачеркнутый» для указанного фрагмента текста.
`TextProperties.allCapitals`	true/false	Значение **true** устанавливает все буквы указанного фрагмента текста как прописные. Значение **false** устанавливает все буквы указанного фрагмента текста как строчные.
`TextProperties.scriptPosition`	Значение таблицы DocumentAPI.ScriptPosition	Устанавливает отображение символа в виде надстрочного, подстрочного знака или в нормальном режиме.
`TextProperties.textColor`	DocumentAPI.ColorRGBA	Цвет указанного фрагмента документа.

Поле	Значение	Описание
TextProperties.backgroundColor	DocumentAPI.ColorRGBA	Цвет фона указанного фрагмента документа.
TextProperties.characterSpacing	Числовое с плавающей точкой, read/write	Размер межсимвольного интервала.

Пример:

```
local props = DocumentAPI.TextProperties()
props.fontName = "XO Oriel"
props.fontSize = 20

-- доступ к тексту третьего абзаца
local text = document:getBlocks():getParagraph(2):getRange()

-- установить свойства фрагмента текста
text:setTextProperties(props)
```

3.30 Таблица DocumentAPI.Range

Таблица **DocumentAPI.Range** предоставляет доступ к указанному фрагменту текстового документа.

3.30.1 Метод Range:getBegin

Метод возвращает позицию в начале фрагмента текстового документа.

Пример:

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getBegin() -- в начало документа
pos:insertText("Привет")
```

3.30.2 Метод Range:getEnd

Метод возвращает позицию в конце фрагмента текстового документа, не включая последний символ paragraph mark.

Пример:

```
local range = document:getRange() -- содержимое всего документа
local pos = range:getEnd() -- в конец документа
pos:insertText("из Lua!")
```

3.30.3 Метод Range:extractText

Метод возвращает содержимое фрагмента в виде строки текста. Находящиеся внутри области изображения, таблицы и пр. игнорируются.

Пример:

```
local range = document:getRange() -- содержимое всего документа
local text    = range:extractText()
print( text ) -- Привет из Lua!
```

3.30.4 Метод Range:removeContent

Метод полностью удаляет содержимое фрагмента текстового документа.

Пример:

```
local range = document:getRange() -- содержимое всего документа
range:removeContent()
```

3.30.5 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример:

```
local range = document:getRange() -- содержимое всего документа
range:replaceText("Новый текст")
```


3.30.6 Метод Range:getTextProperties

Метод возвращает таблицу с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью таблицы **DocumentAPI.TextProperties** (см. раздел 3.29).

Пример:

```
local range = document:getRange()
local props = range:getTextProperties()
props.italic = true
range:setTextProperties(props) -- текстовый фрагмент оформлен курсивом
```

3.30.7 Метод Range:setTextProperties

Метод применяет таблицу с настройками форматирования для фрагмента текстового документа.

Пример:

```
local props = DocumentAPI.TextProperties()
props.italic = true

local range = document:getRange()
local pos = range:getBegin()
range:setTextProperties(props) -- текстовый фрагмент оформлен курсивом
```

3.30.8 Метод Range:enumerateBlocks

Предоставляет возможность итерации по блокам.

Пример:

```
local range = document:getRange()
for block in range:enumerateBlocks() do
    print(block:getRange():extractText())
end
```

3.30.9 Метод Range:enumerateTrackedChanges

Предоставляет возможность итерации по отслеживаемым изменениям.

Пример:

```
local tracked_changes = document:getRange():enumerateTrackedChanges()
```

3.30.10 Метод Range:getComments

Обеспечивает доступ к комментариям в диапазоне.

Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам, если таковые имеются. Если дат нет, то порядок комментариев не определен.

Пример:

```
local comments = document:getRange():getComments()
```


3.30.11 Метод Range:getParagraphs

Обеспечивает доступ к абзацам в диапазоне.

Пример:

```
local paragraphs = document:getRange():getParagraphs()
```


3.31 Таблица DocumentAPI.Table

Таблица **DocumentAPI.Table** предоставляет доступ к листу электронной таблицы или отдельной таблице в составе текстового документа.

3.31.1 Метод Table:setName

Метод устанавливает наименование листа электронной таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:setName( "Первый" )
```

3.31.2 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

Пример:

Напечатать наименование первого листа табличного документа.

Нумерация листов начинается с 0.

```
local tbl = document:getBlocks():getTable( 0 )
print ( tbl:getName () ) -- Первый
```

3.31.3 Метод Table:getRowsCount

Метод позволяет получить количество строк на листе табличного документа.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getRowsCount()) -- 20
```

3.31.4 Метод Table:getColumnsCount

Метод позволяет получить количество столбцов на листе табличного документа.

Пример:

```
local tbl = document:getBlocks():getTable(0)
print(tbl:getColumnsCount()) -- 10
```

3.31.5 Метод Table:getCell

Метод позволяет получить доступ к управлению отдельной ячейкой таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell = tbl:getCell("B2")
print(cell:getFormattedValue())
```


3.31.6 Метод Table:getCellRange

Метод позволяет получить доступ к диапазону ячеек таблицы.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local rng = tbl:getCellRange("A1:C4")
for c in rng:enumerate() do
   print(c:getFormattedValue())
end
```

3.31.7 Метод Table:insertColumnAfter

Метод предназначен для вставки нового столбца после указанной позиции в таблице.

Вызов:

`insertColumnAfter( columnIndex, copyColumnStyle, columnsCount )`, где:

- `columnIndex` – индекс столбца в таблице, после которого производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение **true**, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение **false**, то настройки форматирования не копируются. Значение по умолчанию – **true**.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию – единица.

Пример:

```
-- Создать в документе новую таблицу 2x2
t_id = document:getRange():getBegin():insertTable(2,2,"SomeTable")
t = document:getBlocks():getTable(t_id)

-- Добавление двух столбцов в середину таблицы, без наследования настроек форматирования
tbl:insertColumnAfter(0, false, 2)
```


3.31.8 Метод Table:insertColumnBefore

Метод предназначен для вставки нового столбца до указанной позиции в таблице.

Вызов:

`insertColumnBefore( columnIndex, copyColumnStyle, columnsCount )`, где:

- `columnIndex` – индекс столбца в таблице, перед которым производится вставка. Индексация столбцов начинается с нуля.
- `copyColumnStyle` – флаг наследования стиля. Если этот параметр установлен в значение **true**, то новый столбец наследует настройки форматирования столбца с индексом `columnIndex`. Если параметр `copyColumnStyle` установлен в значение **false**, то настройки форматирования не копируются. Значение по умолчанию – **true**.
- `columnsCount` – количество вставляемых столбцов. Значение по умолчанию – единица.

Пример:

```
-- Создать в документе новую таблицу 2x2
t_id = document:getRange():getBegin():insertTable(2,2,"SomeTable")
t = document:getBlocks():getTable(t_id)

-- Добавление двух столбцов в середину таблицы, без наследования настроек форматирования
tbl:insertColumnBefore(1, false, 2)
```


3.31.9 Метод Table:insertRowAfter

Метод предназначен для вставки новой строки после указанной позиции в таблице.

Вызов:

`insertRowAfter( rowIndex, copyRowStyle, rowsCount )`, где:

- `rowIndex` – индекс строки в таблице, после которой производится вставка. Индексация строк начинается с нуля.
- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение **true**, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение **false**, то настройки форматирования не копируются. Значение по умолчанию – **true**.
- `rowsCount` – количество вставляемых строк. Значение по умолчанию – единица.

Пример:

```
-- Создать в документе новую таблицу 2x2
t_id = document:getRange():getBegin():insertTable(2,2,"SomeTable")
t = document:getBlocks():getTable(t_id)

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowAfter(0, false, 2)
```

3.31.10 Метод Table:insertRowBefore

Метод предназначен для вставки новой строки до указанной позиции в таблице.

Вызов:

`insertRowBefore( rowIndex, copyRowStyle, rowsCount )`, где:

- `rowIndex` – индекс строки в таблице, перед которой производится вставка. Индексация строк начинается с нуля.
- `copyRowStyle` – флаг наследования стиля. Если этот параметр установлен в значение **true**, то новая строка наследует настройки форматирования строки с индексом `rowIndex`. Если параметр `copyRowStyle` установлен в значение **false**, то настройки форматирования не копируются. Значение по умолчанию – **true**.
- `rowsCount` – количество вставляемых строк. Значение по умолчанию – единица.

Пример:

```
-- Создать в документе новую таблицу 2x2
t_id = document:getRange():getBegin():insertTable(2,2,"SomeTable")
t = document:getBlocks():getTable(t_id)

-- Добавление двух строк в середину таблицы, без наследования настроек форматирования
tbl:insertRowBefore(1, false, 2)
```


3.31.11 Метод Table:removeColumn

Метод предназначен для удаления столбца таблицы, начиная с заданного индекса.

Вызов:

`removeColumn(columnIndex, columnsCount)`, где:

- `columnIndex` – индекс столбца, начиная с которого будет удалено `columnsCount` столбцов. Индексация столбцов начинается с нуля.
- `columnsCount` – количество столбцов для удаления. Значение по умолчанию – единица.

3.31.12 Метод Table:removeRow

Метод предназначен для удаления строки таблицы, начиная с заданного индекса.

Вызов:

`removeRow(rowIndex, rowsCount)`, где:

- `rowIndex` – индекс строки, начиная с которого будет удалено `rowsCount` строк. Индексация строк начинается с нуля.
- `rowsCount` – количество строк для удаления. Значение по умолчанию – единица.

3.31.13 Метод Table:setColumnWidth

Метод устанавливает ширину столбца таблицы в пунктах (1/72 дюйма).

Вызов:

`setColumnWidth( columnIndex, width )`, где:

- `columnIndex` – индекс столбца в таблице, для которого устанавливается значение ширины. Индексация столбцов начинается с нуля.
- `width` – ширина столбца в пунктах (1/72 дюйма).

Пример:

```
t_id = document:getRange():getBegin():insertTable(2,2,"SomeTable")
t = document:getBlocks():getTable(t_id)

-- Установить ширину столбца в 400 pt
t:setColumnWidth(1,400)
```


3.31.14 Метод Table:duplicate

Для создания копии листа в табличном документе используется метод **duplicate**. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:duplicate()
```

3.31.15 Метод Table:moveTo

Для перемещения листа таблицы по указанному индексу в табличном документе используется метод **moveTo**. Указанный индекс должен быть меньше или равен количеству листов в документе. Индексация листов начинается с ноля. Метод может быть использован только в табличном документе.

Пример:

```
-- В табличном документе два листа с индексами 0 и 1.
-- Поменяем их местами.
local tbl = document:getBlocks():getTable(0)
tbl:moveTo(1)
```


3.31.16 Объединение и разделение ячеек таблицы

Допустимо объединение произвольного числа ячеек на выбранном листе электронной таблицы. При объединении указанный диапазон становится единой ячейкой. После завершения операции объединенная ячейка получает значение первой ячейки диапазона.

Для объединения нескольких ячеек используйте метод `merge`, принадлежащий таблице **CellRange** (см. раздел 3.28).

Пример:

```
-- Объединение ячеек A1 и A2 на первом листе табличного документа
local tbl = document:getBlocks():getTable(0)
tbl:getCellRange("A1:A2"):merge()
```

Допустимо разъединение только тех ячеек, которые были объединены ранее. После завершения операции данные, содержавшиеся в объединенной ячейке, будут помещены в верхнюю левую ячейку диапазона.

Для разъединения ячеек используйте метод `unmerge`, принадлежащий таблице **Cell** (см. раздел 3.32).

Пример:

```
local tbl = document:getBlocks():getTable(0)
-- Ячейка A1 является результатом объединения диапазона A1:A2
tbl:getCell("A1"):unmerge()
```


3.32 Таблица DocumentAPI.Cell

Таблица **DocumentAPI.Cell** предоставляет доступ к ячейке на листе табличного документа.

3.32.1 Метод Cell:getRange

Метод возвращает объект `Range` для управления содержимым ячейки.

3.32.2 Метод Cell:setBorders

Метод предназначен для установки границ ячейки. См. описание метатаблицы **DocumentAPI.Borders** в разделе 3.26.

3.32.3 Метод Cell:setFormula

Метод позволяет вставить формулу в ячейку табличного документа.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A1"):setNumber(2.3)
tbl:getCell("A2"):setNumber(3.2)
tbl:getCell("A3"):setFormula("=SUM(A1:A2)") -- 5,5
```

3.32.4 Метод Cell:getFormat

Метод возвращает формат ячейки. Список поддерживаемых форматов ячеек см. в разделе 3.11.

3.32.5 Метод Cell:setFormat

Метод устанавливает формат ячейки.

Список поддерживаемых форматов см. в разделе 3.11.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A1"):setNumber(2.3) -- Формат: Общий, отображение 2,3
-- set format
-- Формат: Экспоненциальный, отображение 2,3E+00
tbl:getCell("A1"):setFormat(DocumentAPI.CellFormat_Scientific)
```


3.32.6 Метод Cell:getFormattedValue

Метод позволяет получить значение ячейки в текущем формате.

Список поддерживаемых форматов см. в разделе 3.11.

Пример:

```
local tbl = document:getBlocks():getTable(0)
-- getFormattedValue
print(tbl:getCell("A6"):getFormattedValue()) -- 21.6.1972
```

3.32.7 Метод Cell:setFormattedValue

Анализирует переданное значение и автоматически устанавливает корректный формат ячейки и ее значение. В случае, если распознать тип переданного значения не удается, то для ячейки устанавливается формат "Текстовый" (`DocumentAPI.CellFormat.Text`).

Список поддерживаемых форматов см. в разделе 3.11.

3.32.8 Метод Cell:unmerge

Разъединяет несколько ячеек, которые были объединены ранее.

Пример:

```
local tbl = document:getBlocks():getTable(0)
-- Ячейка A1 является результатом объединения диапазона A1:A2
tbl:getCell("A1"):unmerge()
```

3.32.9 Метод Cell:setContent

Определяет и устанавливает соответствующую формулу/значение, а затем форматирует ячейку. Устанавливает текст, если автоопределение не удалось.

Пример:

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0, 0))
cell:setContent("=A2+A3")
```

3.32.10 Метод Cell:getBorders

Получает границы ячейки.

Пример:

```
local cell = document:getBlocks():getTable(0):getCell(DocumentAPI.CellPosition(0, 0))
local borders = cell:getBorders()
```


3.32.11 Метод Cell:getRawValue

Возвращает значение ячейке в формате «Общий» (без форматирования).

Пример:

```
local tbl = document:getBlocks():getTable(0)
local val = tbl:getCell("A6"):getRawValue()
```

3.32.12 Метод Cell:getCustomFormat

Возвращает строку формата ячейки.

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cust_format = tbl:getCell("A6"):getCustomFormat()
```

3.32.13 Метод Cell:setCustomFormat

Устанавливает формат ячейки.

Пример:

```
local tbl = document:getBlocks():getTable(0)
tbl:getCell("A6"):setCustomFormat("0,00")
```

3.32.14 Метод Cell:setBool

Устанавливает для ячейки значение типа `boolean`.

Пример:

```
local tbl = document:getBlocks():getTable(0)
-- значение ячейки ИСТИНА
tbl:getCell("A6"):setBool(true)
```

3.32.15 Метод Cell:setNumber

Устанавливает для ячейки значение типа «Числовой».

Пример:

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell(DocumentAPI.CellPosition(0, 0))
cell:setNumber(0.0001)
```

3.32.16 Метод Cell:setText

Устанавливает для ячейки значение типа «Текстовый».

Пример:

```
local sheet = document:getBlocks():getTable(0)
local cell = sheet:getCell(DocumentAPI.CellPosition(0, 0))
local trackingChanges = "Disabled"
if document:isChangesTrackingEnabled() then
   trackingChanges = "Enabled"
end
cell:setText(trackingChanges)
```


3.32.17 Метод Cell:getFormulaAsString

Позволяет получить текст формулы в данной ячейке.

Формула – это любое выражение в ячейке, которое начинается со знака равенства (=).

Пример:

```
local tbl = document:getBlocks():getTable(0)
local formula = tbl:getCell("A6"):getFormulaAsString()
```

3.32.18 Метод Cell:getCellProperties

Позволяет получить оформление ячейки (например, фон, вертикальное выравнивание и т. д.).

Пример:

```
local tbl = document:getBlocks():getTable(0)
local cell_props = tbl:getCell("A6"):getCellProperties()
```

3.32.19 Метод Cell:setCellProperties

Позволяет установить оформление ячейки (например, фон, вертикальное выравнивание и т. д.).

Пример:

```
local tbl = document:getBlocks():getTable(0)
local props = tbl:getCell("A6"):getCellProperties()
props.verticalAlignment = DocumentAPI.VerticalAlignment_Center
tbl:getCell("A6"):setCellProperties(props)
```

3.32.20 Метод Cell:getParagraphProperties

Возвращает свойства абзаца, содержащего в ячейке.

3.32.21 Метод Cell:setParagraphProperties

Устанавливает свойства абзаца, содержащего в ячейке.

3.33 Таблица DocumentAPI.Bookmarks

Предоставляет доступ к операциям с закладками (`bookmarks`) в документе. Доступны следующие операции с закладками:

- вставка закладки в указанное местоположение;
- удаление закладки с заданным именем;
- поиск закладки по имени;
- вставка текста в закладку;
- получение текстового содержимого закладки;
- замена текстового содержимого закладки;
- вставка таблицы в закладку;
- удаление содержимого закладки.

3.33.1 Метод Bookmarks:getBookmarkRange

Возвращает объект `Range` для работы с содержимым закладки (`bookmark`) с заданным именем в документе.

Пример:

```
-- В тексте документа происходит замещение содержимого закладки на текст "Lua"
local bms = document:getBookmarks()
local bm_1 = bms:getBookmarkRange( "bm_1" )
bm_1.replaceText("Lua")
```

Вставка закладки в указанное местоположение

```
-- Вставка новой закладки с именем Signers в начало документа
local sig_pos = document:getRange():getBegin()
sig_pos:insertBookmark("Signers")
```

Удаление закладки с заданным именем

```
-- Удаление закладки "Signers"
document:getBookmarks():removeBookmark("Signers")
```

Поиск закладки по имени

```
-- Поиск закладки "Signers" по имени
local sig_rng = document:getBookmarks():getBookmarkRange("Signers")
```

Вставка текста в закладку

```
sig_rng:getBegin():insertText("Лист")
```

Удаление содержимого закладки

```
sig_rng:removeContent()
```


Получение текстового содержимого закладки

```
local msg = sig_rng:extractText()
print(msg)
```

Замена текстового содержимого закладки

```
sig_rng:getBegin():insertText("Лист")
sig_rng:replaceText("Лист согласования")
```

Вставка таблицы в закладку

```
-- Вставка таблицы в закладку "Signers"
local tbl_id = sig_rng:getEnd():insertTable(3,3, "signers_list")
```


3.34 Таблица DocumentAPI.Scripts

Таблица **DocumentAPI.Scripts** предоставляет доступ к операциям управления макрокомандами.

3.34.1 Метод Scripts:getScript

Метод возвращает таблицу **Script** для управления отдельной макрокомандой.

Пример:

```
-- Макрокоманда get_app
local scripts = document:getScripts()
local sc = scripts:getScript("get_app")
print(sc:getName()) --get_app
print(sc:getBody())  --local doc = document:getRange()
```

3.34.2 Метод Scripts:setScript

Метод добавляет макрокоманду `script_name` с кодом `script_code` в текущий документ. Сохранение макрокоманд для последующего использования возможно только при работе с документами формата XODT или ODT и XODS или ODS.

При работе с документами формата XLSX, DOCX, XLS и DOC возможно динамическое добавление и исполнение макрокоманд, однако код макрокоманд не будет сохранен в файл документа.

Пример:

```
-- Создание макрокоманды new_mc
local scripts = document:getScripts()
--
local script_name = "new_mc"
--
local script_code = "local scripts = document:getScripts()\nlocal sc =
scripts:getScript(\"new_mc\")\nprint(sc:getName())\nprint(sc:getBody())"
--
scripts:setScript( script_name, script_code)
```

3.34.3 Метод Scripts:removeScript

Метод удаляет макрокоманду с именем `script_name` из текущего документа.

Пример:

```
-- Удаление макрокоманды new_mc
local scripts = document:getScripts()
scripts:removeScript( "new_mc" )
```


3.35 Таблица DocumentAPI.Script

Таблица **DocumentAPI.Script** предназначена для управления отдельной макрокомандой.

3.35.1 Метод Script:getName

Метод возвращает имя макрокоманды.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("get_app")
print(sc:getName()) -- get_app
```

3.35.2 Метод Script:setName

Метод устанавливает имя для макрокоманды.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("get_app")
sc:setName("new_get_app")
```

3.35.3 Метод Script:getBody

Метод возвращает текст макрокоманды в виде строки.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("get_app")
local code = sc:getBody()
```

3.35.4 Метод Script:setBody

Метод устанавливает текст макрокоманды, полностью заменяя уже имеющийся текст.

Пример:

```
local scripts = document:getScripts()
local sc = scripts:getScript("get_app")
sc:setBody( "local scripts = Application.document:getScripts()\nlocal sc =
scripts:getScript(\"new_mc\")\nprint(sc:getName())\nprint(sc:getBody())")
```


3.36 Таблица DocumentAPI.Search

Таблица **DocumentAPI.Search** предоставляет доступ к механизму поиска и замены фрагментов документа, открытого в редакторе текста или таблиц.

3.36.1 Метод DocumentAPI:createSearch

Метод инициализирует механизм поиска для текущего документа. Возвращает ссылку на таблицу **Search**, с помощью методов которой выполняются поисковые запросы.

3.36.2 Метод Search:findText

Метод выполняет поиск строки `text` без учета регистра во всем документе. Результат возвращается в виде таблицы областей **Range** (см. раздел 3.30), содержащих искомый фрагмент.

Если строка `text` не обнаружена, возвращается пустая таблица.

Пример:

```
search = DocumentAPI.createSearch(document)
ranges = search:findText("English")
for occurrence in ranges do
    print(occurrence:extractText())
end
```


3.37 Таблица DocumentAPI.Position

Таблица **DocumentAPI.Position** представляет определенное местоположение в текстовом документе.

3.37.1 Метод Position:insertText

Метод предназначен для вставки текстовой строки в заданное местоположение в документе.

Пример:

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
begin_pos:insertText("Текст в начале строки")
```

3.37.2 Метод Position:insertTable

Метод предназначен для вставки таблицы с заданным числом строк и столбцов в заданное местоположение в документе. Возвращает уникальный ID таблицы в документе.

Следует учитывать, что при вставке таблицы к ее имени автоматически добавляется порядковый номер, начинающийся с единицы. Таким образом, вызов

```
t_id = begin_pos:insertTable(3,3,"Table")
```

приведет к созданию в документе таблицы с именем “Table1”.

Пример:

```
local rng = document:getRange()
local begin_pos = rng:getBegin()
t_id = begin_pos:insertTable(3,3,"Table")
```

3.37.3 Метод Position:insertPageBreak

Метод предназначен для вставки разрыва страницы в указанную позицию в документе.

Пример:

```
local rng = document:getRange()
local end_pos = rng:getEnd()
end_pos:insertPageBreak()
```

3.37.4 Метод Position:insertLineBreak

Метод предназначен для вставки жесткого перевода строки.

Пример:

```
local rng = document:getRange()
local end_pos = rng:getEnd()
```


3.37.5 Метод Position:insertBookmark

Вставляет закладку с наименованием в текущую позицию.

Пример:

```
document:getRange():getEnd():insertBookmark("Bookmark example")
```

3.37.6 Метод Position:insertSectionBreak

Вставляет в позицию разрыв раздела.

3.37.7 Метод Position:__eq

Позволяет использовать оператор сравнения `==` для определения эквивалентности значений двух местоположений в документе.

3.38 Функция DocumentAPI.createScripting

Функция **DocumentAPI.createScripting** позволяет обеспечить выполнение макрокоманды, хранящейся в документе.

3.38.1 Метод createScripting:runScript

Метод предназначен для запуска макрокоманды, хранящейся в документе.

Пример:

```
scripting = DocumentAPI.createScripting(document)
scripting:runScript("OtherScriptName")
```

где `OtherScriptName` имя макрокоманды, хранящейся в документе.

3.39 Таблица DocumentAPI.Comment

Таблица **DocumentAPI.Comment** представляет доступ к свойствам примененного к диапазону комментария.

3.39.1 Метод Comment:getRange

Метод возвращает объект `Range`, соответствующего объекта `Comment`.

Пример:

```
first_comment_range = comment:getRange()
```

3.39.2 Метод Comment:getText

Метод возвращает текст комментария.

Пример:

```
first_comment_text = comment:getText()
```

3.39.3 Метод Comment:getAudioUrl

Метод возвращает путь к файлу аудиокомментария.

Пример:

```
first_comment_audio_url = comment:getAudioUrl()
```

3.39.4 Метод Comment:getInfo

Метод предоставляет доступ к отслеживаемой информации об изменении комментария (автор изменения, дата и т. д.).

3.40 Таблица DocumentAPI.Comments

Таблица **DocumentAPI.Comments** представляет интерфейс для доступа к коллекции комментариев диапазона.

3.40.1 Метод Comments:enumerate

Метод возвращает коллекцию комментариев диапазона.

Пример:

```
local comments = document:getRange():getComments():enumerate()
```


3.41 Таблица DocumentAPI.TrackedChange

Таблица **DocumentAPI.TrackedChange** представляет интерфейс для отслеживания изменений в документе.

3.41.1 Метод TrackedChange:getRange

Метод возвращает объект `Range`, который соответствует измененному диапазону внутри абзаца.

Пример:

```
tracked_change_range = tracked_change:getRange()
```

3.41.2 Метод TrackedChange:getType

Метод позволяет получить информацию о типе отслеживаемого изменения. Если отслеживаемое изменение добавлено, то метод возвращает ноль, если удалено, то метод возвращает единицу.

Пример:

```
tracked_change_type = tracked_change:getType()
```

3.41.3 Метод TrackedChange:getInfo

Метод позволяет получить информацию об отслеживаемых изменениях.

Пример:

```
tracked_change_info = tracked_change:getInfo()
```


3.42 Таблица DocumentAPI.TrackedChangeInfo

Таблица **DocumentAPI.TrackedChangeInfo** содержит информацию об отслеживаемых изменениях.

Описание полей таблицы **DocumentAPI.TrackedChangeInfo** представлено в Таблица 24.

Таблица 24 – Описание полей таблицы DocumentAPI.TrackedChangeInfo

Поле	Тип	Описание
`DocumentAPI.TrackedChangeInfo.author`	Строка	Автор изменений
`DocumentAPI.TrackedChangeInfo.timeStamp`	Таблица	Дата и время изменений см. раздел 3.44
`DocumentAPI.TrackedChangeInfo.__eq`	Функция	Позволяет использовать оператор сравнения `==` для определения эквивалентности двух отслеживаемых изменений.

3.42.1 Метод TrackedChangeInfo:__eq

Метод позволяет использовать оператор сравнения **==** для определения эквивалентности двух отслеживаемых изменений.

3.43 Таблица DocumentAPI.Paragraphs

Таблица **DocumentAPI.Paragraphs** представляет предоставляет доступ к коллекции абзацев.

3.43.1 Метод Paragraphs:setListSchema

Метод устанавливает тип маркированного или нумерованного списка (см. раздел 3.13).

Пример:

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
```

3.43.2 Метод Paragraphs:setListLevel

Метод устанавливает глубину вложенности элемента списка.

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:setListLevel(1)
```

3.43.3 Метод Paragraphs:increaseListLevel

Метод увеличивает уровень списка на единицу. В случае, если максимальный уровень уже установлен, увеличения не происходит.

Пример:

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:increaseListLevel()
```

3.43.4 Метод Paragraphs:decreaseListLevel

Метод уменьшает уровень списка на единицу. В случае если минимальный уровень уже установлен, уменьшения не происходит.

Пример:

```
local paragraphs = document:getRange():getParagraphs()
paragraphs:setListSchema(DocumentAPI.ListSchema_BulletCircleSolid)
paragraphs:decreaseListLevel()
```

3.43.5 Метод Paragraphs:enumerate

Метод возвращает коллекцию абзацев.

Пример:

```
local paragraphs = document:getRange():getParagraphs():enumerate()
```


3.44 Таблица DocumentAPI.DateTime

Таблица **DocumentAPI.DateTime** предоставляет дату и время с точностью до секунды. Описание полей таблицы **DocumentAPI.DateTime** представлено в Таблица 25.

Таблица 25 – Описание полей таблицы DocumentAPI.DateTime

Поле	Тип	Описание
`DocumentAPI.DateTime.year`	Дата	год
`DocumentAPI.DateTime.month`	Дата	месяц
`DocumentAPI.DateTime.day`	Дата	день
`DocumentAPI.DateTime.hour`	Время	часы
`DocumentAPI.DateTime.minute`	Время	минуты
`DocumentAPI.DateTime.second`	Время	секунды
`DocumentAPI.DateTime.__eq`	Функция	Позволяет использовать оператор сравнения == для определения эквивалентности двух значений времени.

3.44.1 Метод DocumentAPI.DateTime:__eq

Метод позволяет использовать оператор сравнения **==** для определения эквивалентности двух значений времени.

3.45 Таблица DocumentAPI.Section

Таблица **DocumentAPI.Section** представляет собой раздел в документе.

3.45.1 Метод Section:setPageProperties

Метод устанавливает высоту и ширину страниц раздела.

3.45.2 Метод Section:getPageProperties

Метод возвращает размеры высоты и ширины для страниц раздела.

3.45.3 Метод Section:setPageOrientation

Метод задает ориентацию страниц раздела.

3.45.4 Метод Section:getPageOrientation

Метод возвращает ориентацию страниц раздела.

3.45.5 Метод Section:getRange

Метод возвращает диапазон в документе, соответствующий данному разделу.

3.45.6 Метод Section:getHeaders

Метод возвращает коллекцию верхних колонтитулов данного раздела.

3.45.7 Метод Section:getFooters

Метод возвращает коллекцию нижних колонтитулов данного раздела.

3.46 Таблица DocumentAPI.PageProperties

Таблица **DocumentAPI.PageProperties** предоставляет размеры страницы. Описание полей таблицы **DocumentAPI.PageProperties** представлено в Таблица 26.

Таблица 26 – Описание полей таблицы DocumentAPI.PageProperties

Поле	Описание
`DocumentAPI.PageProperties.height`	Высота страницы
`DocumentAPI.PageProperties.width`	Ширина страницы

3.47 Таблица DocumentAPI.HeaderFooter

Таблица **DocumentAPI.HeaderFooter** определяет колонтитул документа.

3.47.1 Метод HeaderFooter:getType

Метод предоставляет информацию о типе колонтитула.

3.47.2 Метод HeaderFooter:getBlocks

Метод предоставляет доступ к блокам, которые содержатся в колонтитуле.

3.47.3 Метод HeaderFooter:getRange

Метод предоставляет диапазон с содержанием верхнего или нижнего колонтитулов.

3.48 Таблица DocumentAPI.HeadersFooters

Таблица **DocumentAPI.HeadersFooters** представляет коллекцию верхних и нижних колонтитулов разделе (section). См. описание таблицы **DocumentAPI.Section** в разделе 3.45.

3.48.1 Метод HeadersFooters:Enumerate

Метод возвращает коллекцию колонтитулов.

4 СПРАВОЧНИК ФУНКЦИЙ ГЛОБАЛЬНОЙ ТАБЛИЦЫ EDITORAPI

Глобальная таблица **EditorAPI** содержит функции доступа к внешней функциональности редактора.

4.1 Функция EditorAPI.getSelection

Функция **EditorAPI.getSelection** предоставляет доступ к выделенному фрагменту документа.

В открытом документе может быть выделен только один фрагмент, сколь угодно большого размера.

При использовании в редакторе текста функция **EditorAPI.getSelection** возвращает `Range`, а при использовании в редакторе таблиц функция **EditorAPI.getSelection** возвращает `CellRange`.

Примеры:

Использование функции **EditorAPI.getSelection** в редакторе текста для печати выделенного фрагмента текста.

```
local text = EditorAPI.getSelection():extractText()
print(text)
```

Использование функции **EditorAPI.getSelection** в редакторе таблиц для печати значений ячеек в выделенном фрагменте таблицы.

```
for cell in EditorAPI.getSelection():enumerate() do
    print(cell:getFormattedValue())
end
```


4.2 Функция EditorAPI.messageBox

Функция **EditorAPI.messageBox** выводит на экран сообщение с заданным текстом и отображением кнопки **ОК**, при этом исполнение макрокоманды приостанавливается до нажатия кнопки **ОК**.

Вызов:

```
messageBox(prompt : string)
messageBox(prompt : string)
messageBox(prompt : string, title : string)
```

где:

- `prompt` – текст сообщения;
- `title` – заголовок окна сообщения.

Пример:

```
EditorAPI.messageBox("message", "title")
```


4.3 Функция EditorAPI.showPrintDialog

Функция **EditorAPI.showPrintDialog** показывает стандартное окно печати редактора и распечатывает документ, если пользователь подтверждает необходимость печати.

Значения, возвращаемые функцией **EditorAPI.showPrintDialog**, перечислены в разделе 3.16.

Пример:

```
local result = EditorAPI.showPrintDialog()
print (result)
```

