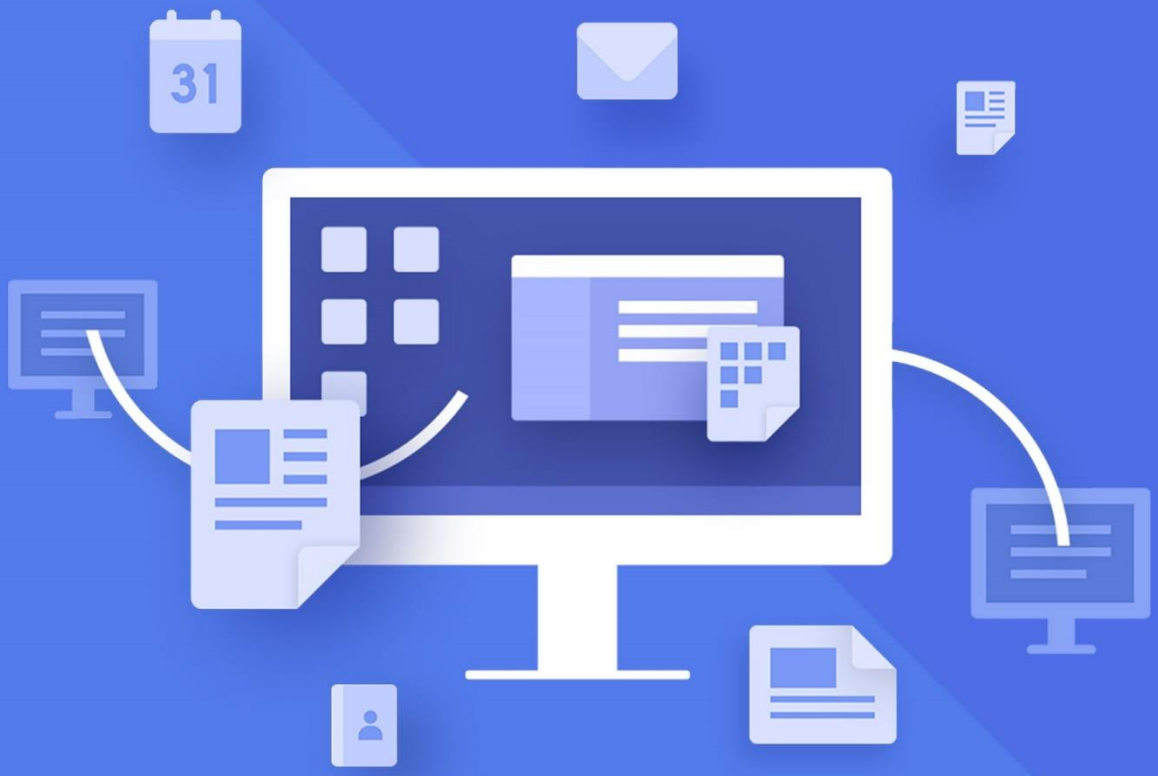


МойОфис®
Стандартный

Справочник макрокоманд

ДОКУМЕНТЫ

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

МОЙОФИС СТАНДАРТНЫЙ

ДОКУМЕНТЫ

СПРАВОЧНИК МАКРОКОМАНД НА ЯЗЫКЕ LUA

2019.03

На 53 листах

Москва

2019

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. Документы
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 – продукт для работы с офисными приложениями в государственных организациях и крупных коммерческих предприятиях. Включает редакторы текста, таблиц, презентаций и приложения для управления почтой, календарем и контактами.

МойОфис Текст обеспечивает удобное и быстрое создание документов с использованием шаблонов, стилей и средств форматирования текста. Функции совместного редактирования[1] обеспечивают эффективную совместную работу сотрудников.

МойОфис Таблица – это приложение для быстрой и удобной работы с электронными таблицами и анализа данных. Продукт поддерживает расширенный набор формул и средств для обработки данных. Совместное редактирование на любой из поддерживаемых платформ обеспечивает быстрый анализ и подготовку документов группой сотрудников даже вне офиса.

МойОфис Презентация – приложение с полным набором инструментов для просмотра графических презентаций.

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.

[1]Функции совместного редактирования доступны при наличии серверной части МойОфис Профессиональный.

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 XODS.

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**.

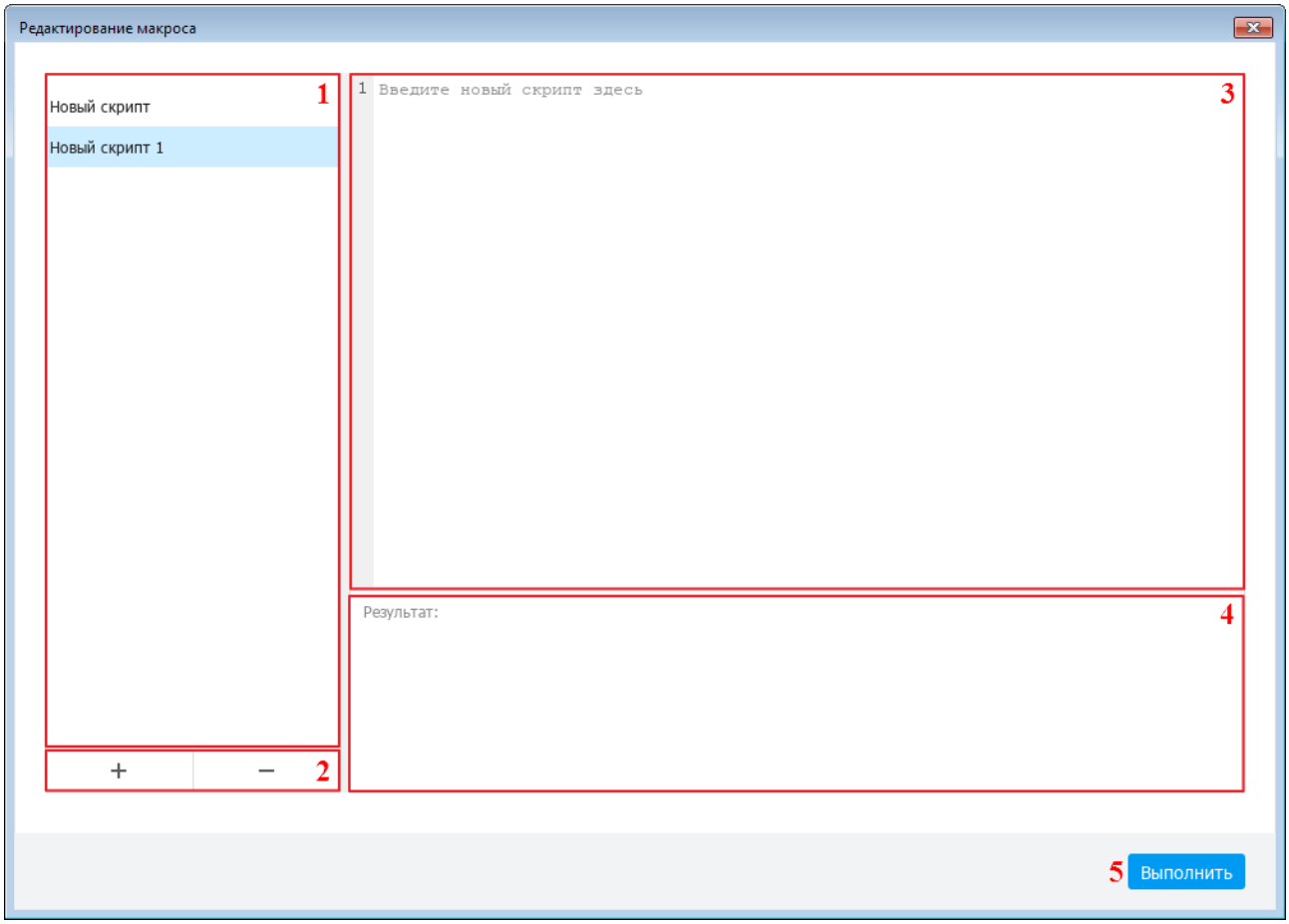


Рисунок 1 – Окно редактора макрокоманд

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 для текстовых документов или XODS 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 3.5) представляет текущий открытый текстовый документ.

Таблица **Range** (см. раздел 3.19) 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 3.18) 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 Таблица CoreAPI.Blocks

3.1.1 Метод Blocks:getParagraph

Возвращает абзац с указанным индексом.

Пример:

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

3.1.2 Метод Blocks:getTable

Для табличного документа возвращает лист (worksheet) с указанным номером. Нумерация листов начинается с 0.

Для текстового документа возвращает таблицу с указанным порядковым номером. Нумерация таблиц начинается с 0.

Пример:

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

3.1.3 Метод Blocks:enumerate

Предоставляет возможность итерирования по всем блокам.

Пример:

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

3.1.4 Метод Blocks:enumerateTables

Предоставляет возможность итерирования по всем таблицам.

Пример:

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

3.1.5 Метод Blocks:enumerateParagraphs

Предоставляет возможность итерирования по всем параграфам.

Пример:

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


3.2 Таблица CoreAPI.VerticalAlignment

При работе в табличном редакторе с помощью таблицы **CoreAPI.VerticalAlignment** осуществляется выравнивание текста, расположенного в ячейках вертикально. Описание полей таблицы **CoreAPI.VerticalAlignment** представлено в таблице 2.

Таблица 2 – Описание полей таблицы CoreAPI.VerticalAlignment

Поле	Свойства	Представление в интерфейсе
CoreAPI.VerticalAlignment_Bottom	[ReadOnly]	
CoreAPI.VerticalAlignment_Center	[ReadOnly]	
CoreAPI.VerticalAlignment_Top	[ReadOnly]	

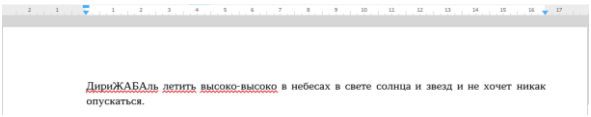
Пример:

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

3.3 Таблица CoreAPI.Alignment

Таблица **CoreAPI.Alignment** предназначена для выравнивания по горизонтали текстовых фрагментов в текстовом редакторе или содержимого ячеек в табличном редакторе. Описание полей таблицы **CoreAPI.Alignment** представлено в таблице 3.

Таблица 3 – Описание полей таблицы CoreAPI.Alignment

Поле	Свойства	Описание
CoreAPI.Alignment_Default	[ReadOnly]	
CoreAPI.Alignment_Left	[ReadOnly]	по левому краю 
CoreAPI.Alignment_Center	[ReadOnly]	по центру 
CoreAPI.Alignment_Right	[ReadOnly]	по правому краю 
CoreAPI.Alignment_Justify	[ReadOnly]	по ширине 

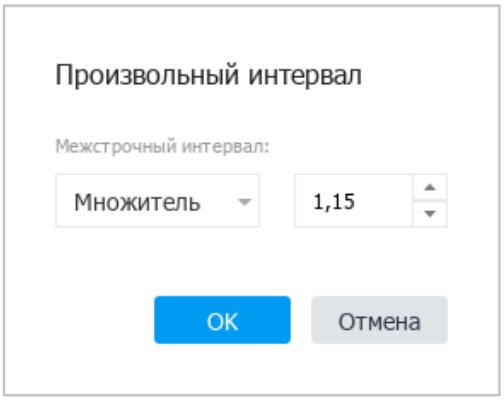
Пример:

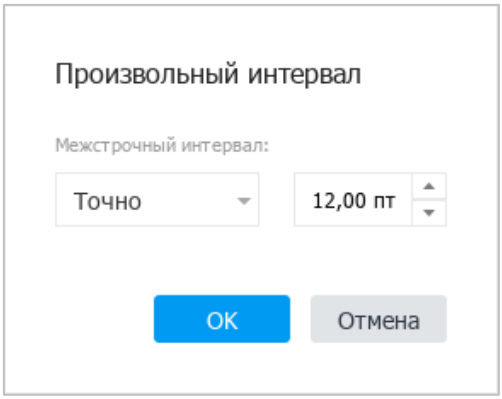
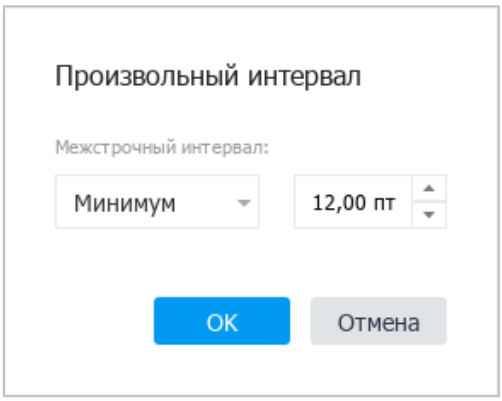
```
local para = document:getBlocks():getParagraph(0)
local props = para:getParagraphProperties()
props.alignment = CoreAPI.Alignment_Center
para:setParagraphProperties(props)
```

3.4 Таблица CoreAPI.LineSpacing

При работе в текстовом редакторе с помощью таблицы **CoreAPI.LineSpacing** осуществляется настройка межстрочного интервала для абзаца текста. Описание полей таблицы **CoreAPI.LineSpacing** представлено в таблице 4.

Таблица 4 – Описание полей таблицы CoreAPI.LineSpacing

Поле	Описание
CoreAPI.LineSpacingRule_Multiple	Установка произвольного межстрочного интервала с использованием множителя. При вызове необходимо указать значение множителя, например: `pPr.lineSpacing = CoreAPI.LineSpacing(1.15, CoreAPI.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 «МойОфис Стандартный. Руководство пользователя»). 

Поле	Описание
CoreAPI.LineSpacingRule_Exact	Установка произвольного межстрочного интервала с использованием точного значения. При вызове необходимо указать точное значение, например: `pPr.lineSpacing = CoreAPI.LineSpacing(12.0, CoreAPI.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 «МойОфис Стандартный. Руководство пользователя»). 
CoreAPI.LineSpacingRule_AtLeast	Установка произвольного междустрочного интервала с использованием минимального значения. При вызове необходимо указать минимальное значение, например: `pPr.lineSpacing = CoreAPI.LineSpacing(12.0, CoreAPI.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 «МойОфис Стандартный. Руководство пользователя»). 

Пример:

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

3.5 Таблица CoreAPI.Document

Таблица **CoreAPI.Document** предоставляет доступ к содержимому открытого текстового или табличного документа. Реализуется через доступ к содержимому документа через сущность `document`. К примеру,

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

предоставляет доступ к первому абзацу текстового документа.

3.5.1 Метод document:getBlocks

Метод предоставляет доступ к метатаблице **Blocks** (см. раздел 3.1) и далее к отдельным фрагментам (абзацам, таблицам и т.д.), из которых составлен документ.

Пример:

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

3.5.2 Метод document:getBookmarks

Метод предоставляет доступ к таблице закладок **Bookmarks** (см. раздел 3.22).

Пример:

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

3.5.3 Метод document:getScripts

Метод предоставляет доступ к таблице макрокоманд **Scripts** (см. раздел 3.24), содержащихся в документе.

Пример:

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

3.5.4 Метод document:getRange

Метод предоставляет доступ ко всему документу как области данных.

Пример:

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

3.6 Таблица CoreAPI.Paragraph

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

3.6.1 Метод Paragraph:getParagraphProperties

Метод предоставляет доступ к таблице свойств форматирования абзаца **ParagraphProperties** (см. раздел 3.7), таким как выравнивание текста, межстрочные интервалы, отступы и т.п.

3.6.2 Метод Paragraph:setParagraphProperties

Метод предназначен для обновления таблицы свойств форматирования абзаца **ParagraphProperties** (см. раздел 3.7).

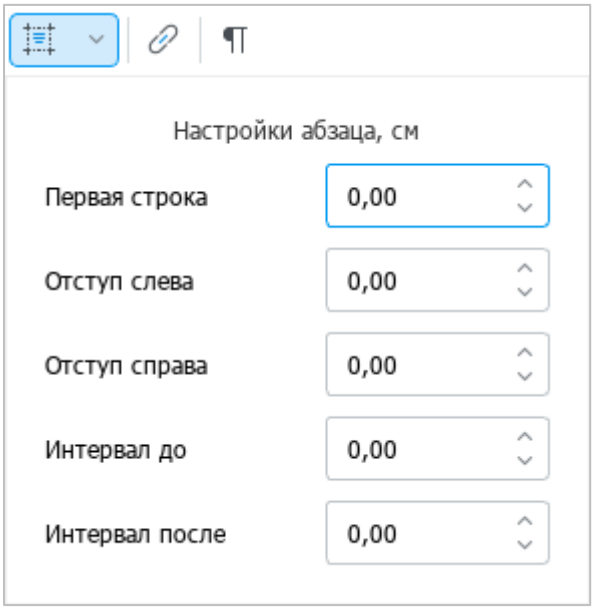
Пример:

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


3.7 Таблица CoreAPI.ParagraphProperties

Таблица **CoreAPI.ParagraphProperties** предназначена для управления свойствами форматирования абзаца при работе в текстовом редакторе. Описание полей таблицы **CoreAPI.ParagraphProperties** представлено в таблице 5.

Таблица 5 – Описание полей таблицы CoreAPI.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, см**, в поле **Интервал после** (подробнее см. в документе «МойОфис Стандартный. 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** (см. рисунок выше), в поле **Интервал до** (подробнее см. в документе «МойОфис Стандартный. Руководство пользователя»).
ParagraphProperties.alignment	[ReadWrite]	Выравнивание текстового фрагмента по горизонтали. Список допустимых значений в таблице **CoreAPI.Alignment** (см. раздел 3.3).

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** (см. рисунок выше), в поле **Первая строка** (подробнее см. в документе «МойОфис Стандартный. 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** (см. рисунок выше), в поле **Отступ слева** (подробнее см. в документе «МойОфис Стандартный. Руководство пользователя»).
ParagraphProperties.lineSpacing	[ReadWrite]	Расстояние между строк одного абзаца (межстрочный интервал). Для управления значением межстрочного интервала используйте таблицу **CoreAPI.LineSpacing** (см. раздел 3.4).
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** (см. рисунок выше), в поле **Отступ справа** (подробнее см. в документе «МойОфис Стандартный. 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 = CoreAPI.Alignment_Center
para_props.firstLineIndent = 28.3 – значение соответствует 1 см
para_props.leftIndent = 28.3 – значение соответствует 1см
para_props.lineSpacing = CoreAPI.LineSpacing(5.0, CoreAPI.LineSpacingRule_Multiple)
para_props.rightIndent = 28.3 – значение соответствует 1см
--
para:setParagraphProperties(para_props)
```

3.8 Таблица CoreAPI.CellFormat

При работе в табличном редакторе с помощью таблицы **CoreAPI.CellFormat** настраивается формат ячеек. По умолчанию при создании документа всем ячейкам присваивается формат «Общий».

Описание полей таблицы **CoreAPI.CellFormat** представлено в таблице 6.

Таблица 6 – Описание полей таблицы CoreAPI.CellFormat

Поле	Свойства	Описание
CoreAPI.CellFormat_General	[ReadOnly]	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.
CoreAPI.CellFormat_Percentage	[ReadOnly]	Формат ячейки «Процентный». Этот формат используется для представления чисел как процентов. При применении формата «Процентный» введенное число умножается на 100 и обозначается знаком «%».
CoreAPI.CellFormat_Number	[ReadOnly]	Формат ячейки «Числовой». Если в ячейке с форматом «Числовой» содержится дробное число, то можно указать количество знаков, отображаемых в данном числе после разделителя.
CoreAPI.CellFormat_Text	[ReadOnly]	Формат ячейки «Текстовый».
CoreAPI.CellFormat_Currency	[ReadOnly]	Формат ячейки «Денежный». Этот формат используется для представления чисел со знаком или кодом валюты.
CoreAPI.CellFormat_Accounting	[ReadOnly]	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.

Поле	Свойства	Описание
CoreAPI.CellFormat_Date	[ReadOnly]	Формат ячейки «Дата». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ДД.ММ.ГГГГ.
CoreAPI.CellFormat_Time	[ReadOnly]	Формат ячейки «Время». Этот формат автоматически присваивается числам, введенным в определенном виде, например, ЧЧ:ММ.
CoreAPI.CellFormat_Fraction	[ReadOnly]	Формат ячейки «Дробный». Этот формат используется для представления дробных чисел в виде обыкновенных дробей (то есть дробная часть заменяется на числитель и знаменатель).
CoreAPI.CellFormat_Scientific	[ReadOnly]	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(CoreAPI.CellFormat_General)
local cell_B2 = tbl:getCell("B2")
cell_B2:setFormat(CoreAPI.CellFormat_Percentage)
local cell_B3 = tbl:getCell("B3")
cell_B3:setFormat(CoreAPI.CellFormat_Number)
```

Результат:

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

3.9 Таблица CoreAPI.TextLayout

При работе в табличном редакторе с помощью таблицы **CoreAPI.TextLayout** настраивается режим отображения длинного текста в ячейках. Описание полей таблицы **CoreAPI.TextLayout** представлено в таблице 7.

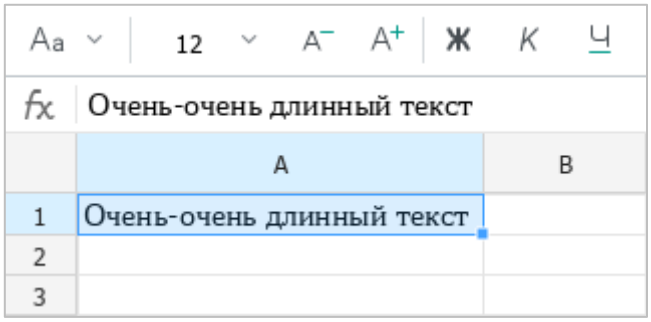
Таблица 7 – Описание полей таблицы CoreAPI.TextLayout

Поле	Свойства	Описание
CoreAPI.TextLayout_SingleLine	[ReadWrite]	Отображение длинного текста в одну строку с наложением на соседние ячейки.
CoreAPI.TextLayout_WrapByWords	[ReadWrite]	Перенос данных внутри ячейки по словам.
CoreAPI.TextLayout_ShrinkSizeToFitWidth	[ReadWrite]	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 = CoreAPI.TextLayout_ShrinkSizeToFitWidth
cell_A1:setCellProperties(props)
```

Результат:



3.10 Таблица CoreAPI.CellPosition

Таблица **CoreAPI.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 ( CoreAPI.CellPosition ( 2, 0 ) ) – ячейка A3
```

либо

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

3.10.1 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 = CoreAPI.CellPosition(0,0)
print( pos:toString() ) – (row: 0, column: 0)
```

3.10.1.1 Поле CellPosition.row

Номер строки в позиции ячейки. Нумерация строк начинается с 0.

3.10.1.2 Поле CellPosition.column

Номер столбца в значении ячейки. Нумерация столбцов начинается с 0.

3.11 Таблица CoreAPI.ColorRGBA

Таблица **CoreAPI.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 **CoreAPI.ColorRGBA** представлено в таблице 8.

Таблица 8 – Описание таблицы CoreAPI.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 = CoreAPI.LineProperties()
line_prop.color = CoreAPI.ColorRGBA(55, 146, 179, 200)
```


3.12 Таблица CoreAPI.CellProperties

Таблица **CoreAPI.CellProperties** предназначена для выравнивания содержимого в ячейках таблицы. Описание полей таблицы **CoreAPI.CellProperties** представлено в таблице 9.

Таблица 9 – Описание полей таблицы CoreAPI.CellProperties

Поле	Свойства	Значение
CellProperties.verticalAlignment	[ReadWrite]	Настройка вертикального выравнивания значения ячейки. См. описание таблицы **CoreAPI.Alignment** в разделе 3.3.
CellProperties.backgroundColor	[ReadWrite]	Настройка цвета фона ячейки. См. описание таблицы **CoreAPI.ColorRGBA** в разделе 3.11.
CellProperties.textLayout	[ReadWrite]	Настройка вариантов отображения значения ячейки. См. описание таблицы **CoreAPI.TextLayout** в разделе 3.9.

Пример:

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


3.13 Таблица CoreAPI.LineProperties

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

3.13.1 Поле LineProperties.style

Поле предназначено для установки типа линии. Допустимые значения представлены в таблице 10.

Таблица 10 – Типы линии

Значение	Описание
CoreAPI.LineStyle_NoLine	Граница отсутствует
CoreAPI.LineStyle_Solid	
CoreAPI.LineStyle_Dot	
CoreAPI.LineStyle_Dash	
CoreAPI.LineStyle_LongDash	
CoreAPI.LineStyle_DashDot	
CoreAPI.LineStyle_DotDotDash	
CoreAPI.LineStyle_Double	

3.13.2 Поле LineProperties.width

Поле предназначено для установки ширины линии.

3.13.3 Поле LineProperties.color

Поле предназначено для установки цвета линии.

Пример:

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

3.14 Таблица CoreAPI.Borders

Таблица **CoreAPI.Borders** предназначена для оформления границ отдельной ячейки таблицы. Параметры линии, такие как тип линии, ее ширина и цвет, задаются с помощью таблицы **CoreAPI.LineProperties** (см. раздел 3.13).

Таблица 11 – Методы

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 = CoreAPI.LineProperties()
line_prop.style = CoreAPI.LineStyle_Dash
line_prop.width = 1.5
line_prop.color = CoreAPI.ColorRGBA(55, 146, 179, 200)
--
rb = CoreAPI.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:

3.15 Таблица CoreAPI.RangeBorders

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

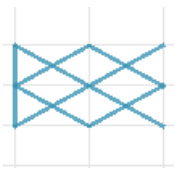
Таблица 12 – Методы

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 = CoreAPI.LineProperties()
line_prop.style = CoreAPI.LineStyle_Solid
line_prop.width = 1.5
line_prop.color = CoreAPI.ColorRGBA(55, 146, 179, 200)
--
local tbl = document:getBlocks():getTable(0)
local cell_rng = tbl:getCellRange("C3:D4")
--
local rb = CoreAPI.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:



3.16 Таблица CoreAPI.CellRange

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

3.16.1 Метод CellRange:enumerate

Метод возвращает коллекцию ячеек.

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
```

3.16.2 Метод CellRange:getBeginRow

Метод возвращает индекс строки первой ячейки диапазона. Нумерация строк начинается с 0.

Пример:

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

3.16.3 Метод CellRange:getBeginColumn

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

Пример:

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

3.16.4 Метод CellRange:getLastRow

Метод возвращает индекс строки последней ячейки диапазона. Нумерация строк начинается с 0.

Пример:

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


3.16.5 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
```

3.16.6 Метод CellRange:setBorders

Метод предназначен для установки границ диапазона ячеек. Отдельные границы устанавливаются с помощью методов таблицы **RangeBorders** (см. раздел 3.15).

3.16.7 Метод CellRange:getCellProperties

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

3.16.8 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 = CoreAPI.CellProperties()
props.backgroundColor = CoreAPI.ColorRGBA(55, 146, 179, 200)
rng:setCellProperties(props)
```

3.16.9 Метод merge

Метод объединяет указанный диапазон ячеек в одну.

Пример:

--Объединение ячеек A1 и A2 на первом листе табличного документа

```
local tbl = document:getBlocks():getTable(0)
tbl:getCellRange("A1:A2"):merge()
```


3.17 Таблица CoreAPI.ScriptPosition

Таблица **CoreAPI.ScriptPosition** предназначена для представления текста в виде надстрочных или подстрочных знаков при работе в текстовом редакторе. Описание полей таблицы **CoreAPI.ScriptPosition** представлено в таблице 13.

Таблица 13 – Поля таблицы CoreAPI.ScriptPosition

Поле	Описание
CoreAPI.ScriptPosition_SuperScript	Надстрочный знак (верхний индекс)
CoreAPI.ScriptPosition_SubScript	Подстрочный знак (нижний индекс)
CoreAPI.ScriptPosition_NoramlScript	Без указания индекса

3.18 Таблица CoreAPI.TextProperties

Таблица **CoreAPI.TextProperties** предназначена для форматирования текста. Описание полей таблицы **CoreAPI.TextProperties** представлено в таблице 14.

Таблица 14 – Описание полей таблицы CoreAPI.TextProperties

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

Пример:

```
local props = CoreAPI.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)
```

3.19 Таблица CoreAPI.Range

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

3.19.1 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")
```

3.19.2 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!")
```

3.19.3 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!
```


3.19.4 Метод Range:removeContent

Метод полностью удаляет содержимое фрагмента текстового документа.

Пример:

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

3.19.5 Метод Range:replaceText

Метод заменяет содержимое фрагмента на указанный текст.

Пример:

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

3.19.6 Метод Range:getTextProperties

Метод возвращает таблицу с текущими настройками форматирования для фрагмента текстового документа. Описание настроек форматирования осуществляется с помощью таблицы **CoreAPI.TextProperties** (см. раздел 3.18).

Пример:

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

3.19.7 Метод Range:setTextProperties

Метод применяет таблицу с настройками форматирования для фрагмента текстового документа.

Пример:

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

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

3.19.8 Метод Range:enumerateBlocks

Предоставляет возможность итерации по блокам.

Пример:

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


3.20 Таблица CoreAPI.Table

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

3.20.1 Метод Table:setName

Метод устанавливает наименование листа электронной таблицы.

Пример:

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

3.20.2 Метод Table:getName

Метод позволяет получить наименование листа табличного документа.

Пример:

Напечатать наименование первого листа табличного документа. Нумерация листов начинается с 0.

Пример:

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

3.20.3 Метод Table:getRowsCount

Метод позволяет получить количество строк на листе табличного документа.

Пример:

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

3.20.4 Метод Table:getColumnsCount

Метод позволяет получить количество столбцов на листе табличного документа.

Пример:

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

3.20.5 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())
```


3.20.6 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
```

3.20.7 Метод 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)
```


3.20.8 Метод 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)
```

3.20.9 Метод 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)
```

3.20.10 Метод 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)
```

3.20.11 Метод 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.

3.20.12 Метод 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.

3.20.13 Метод 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)
```

3.20.14 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 3.16).

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 3.21).

Пример:

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


3.21 Таблица CoreAPI.Cell

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

3.21.1 Метод Cell:getRange

Метод возвращает объект `Range` для управления содержимым ячейки.

3.21.2 Метод Cell:setBorders

Метод предназначен для установки границ ячейки. См. описание метатаблицы **CoreAPI.Borders** в разделе 3.14.

3.21.3 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
```

3.21.4 Метод Cell:getFormat

Метод возвращает формат ячейки. Поддерживаются следующие форматы ячеек:

Общий	CoreAPI.CellFormat.General	12
Процентный	CoreAPI.CellFormat.Percentage	1200,00%
Числовой	CoreAPI.CellFormat.Number	12,00
Денежный	CoreAPI.CellFormat.Currency	12 000,00₽
Финансовый	CoreAPI.CellFormat.Accounting	12 000,00₽
Дата	CoreAPI.CellFormat.Date	11.1.1900
Время	CoreAPI.CellFormat.Time	0:00:00
Дробный	CoreAPI.CellFormat.Fraction	12
Экспоненциальный	CoreAPI.CellFormat.Scientific	1,2E+01
Текстовый	CoreAPI.CellFormat.Text	12

3.21.5 Метод Cell:setFormat

Метод устанавливает формат ячейки. Список поддерживаемых форматов см. в разделе 3.21.4.

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(CoreAPI.CellFormat_Scientific)
```

3.21.6 Метод Cell:getFormattedValue

Метод позволяет получить значение ячейки в текущем формате. Список поддерживаемых форматов см. в разделе 3.21.4.

Пример:

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

3.21.7 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" (CoreAPI.CellFormat.Text). Список поддерживаемых форматов см. в описании Cell:getFormat.

3.21.8 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()
```

3.21.9 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(CoreAPI.CellPosition(0, 0))
cell:setContent("=A2+A3")
```


3.21.10 Метод Cell:getBorders

Получает границы ячейки.

Пример:

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

3.22 Таблица CoreAPI.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.

3.22.1 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")
```


3.23 Таблица CoreAPI.Scripts

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

3.23.1 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()
```

3.23.2 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 XODS.

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)
```

3.23.3 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" )
```

3.24 Таблица CoreAPI.Script

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

3.24.1 Метод Script:getName

Метод возвращает имя макрокоманды.

Пример:

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

3.24.2 Метод Script:setName

Метод устанавливает имя для макрокоманды.

Пример:

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

3.24.3 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()
```

3.24.4 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())")
```


3.25 Таблица CoreAPI.Search

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

3.25.1 Метод CoreAPI:createSearch

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

3.25.2 Метод Search:findText

Метод выполняет поиск строки `text` без учета регистра во всем документе. Результат возвращается в виде таблицы областей **Range** (см. раздел 3.19), содержащих искомый фрагмент. Если строка `text` не обнаружена, возвращается пустая таблица.

Пример:

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


3.26 Таблица CoreAPI.Position

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

3.26.1 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")
```

3.26.2 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")
```

3.26.3 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()
```

3.26.4 Метод Position:insertLineBreak

Метод предназначен для вставки жесткого перевода строки.

Пример:

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