Содержание

Введение

Данная книга является описанием встроенного языка системы 1С:Предприятие и предназначена для специалистов, выполняющих конфигурирование системы для решения конкретной задачи автоматизации учета.

1С:Предприятие является гибкой настраиваемой системой, с помощью которой можно решать широкий круг задач в сфере автоматизации деятельности предприятий. Специфические алгоритмы конфигурации описываются в системе 1С:Предприятие при помощи программной компоненты Конфигуратор (далее по тексту — конфигуратор) в программных модулях, содержащих тексты на встроенном языке системы 1С:Предприятие.

Назначение и краткая характеристика встроенного языка

Встроенный язык системы 1С:Предприятие предназначен для описания (на стадии разработки конфигурации) алгоритмов функционирования прикладной задачи.

Встроенный язык (далее по тексту — язык) представляет собой предметно-ориентированный язык программирования, специально разработанный с учетом возможности его применения не только профессиональными программистами. В частности, все операторы языка имеют как русское, так и англоязычное написание, которые можно использовать одновременно в одном исходном тексте. Основной язык, описываемый в данной книге — русский, однако для каждого оператора языка приводится его англоязычный синоним.

При своей относительной простоте язык обладает некоторыми объектно-ориентированными возможностями, например, правила доступа к атрибутам и методам специализированных типов данных (документам, справочникам и т. п.) подобны свойствам и методам объектов, используемых в других объектно-ориентированных языках. Однако специализированные типы данных не могут определяться средствами самого языка, а задаются в визуальном режиме конфигуратора.

Типизация переменных в языке не жесткая, т. е. тип переменной определяется ее значением. Переменные не обязательно объявлять в явном виде. Неявным определением переменной является ее первое упоминание в левой части оператора присваивания. Возможно также явное объявление переменных при помощи соответствующего оператора. Допускается применение массивов.

Построение книги

Элементы языка в данной книге описываются логически сгруппированными на основании их функциональной направленности, поэтому оглавление данной книги не упорядочено по алфавиту. Обычно в начале главы приводится тип обрабатываемых данных, затем следуют его атрибуты, а в конце исполняемые методы, использующие этот тип данных.

Формат описания элементов языка

Каждый элемент (конструкция) языка, упомянутый в этом руководстве, печатается таким шрифтом. Информация по компонентам языка приводится в виде синтаксической диаграммы, подробного описания и примера исходного текста.

Соглашения и обозначения, принятые в синтаксических диаграммах

В синтаксических диаграммах используются следующие символы:

Символ	Значение
[ ]	В квадратных скобках заключаются необязательные синтаксические элементы.
( )	Круглые скобки заключают в себе список параметров.
|	Вертикальной линией разделяются синтаксические элементы, среди которых нужно выбрать только один.

Синтаксическая диаграмма описания элемента языка

Формат описания элемента языка, используемый в данном руководстве, иллюстрируется синтаксической диаграммой, приведенной ниже.

ЭлементЯзыка

Краткое описание того, что делает данный `ЭлементЯзыка`.

Синтаксис:

```
ЭлементЯзыка(<Параметр1>, <Параметр2>, ...) [ДобКлючевоеСлово]
```

Англоязычный Синтаксис: (в случае языковых конструкций)

```
Keyword(<Параметр1>, <Параметр2>, ...)[AddKeyWord]
```

Англоязычный синоним: (в случае описания методов, функций и процедур)

```
Keyword
```

Параметры:

`<Параметр1>`	краткое описание `<Параметра1>`.
`<Параметр2>`	краткое описание `<Параметра2>`.
`[ДобКлючевоеСлово]`	краткое описание `ДобКлючевоеСлово`.

Возвращаемое значение:
Тип и краткое описание возвращаемого значения.

Описание:

Подробное описание того, что реализует `ЭлементЯзыка`.

Пример:

- Краткое описание примера

```
// Исходный текст примера
```

См. также: Ссылки на другие методы, процедуры или функции.

Глава 1
Формат исходных текстов программных модулей

Что такое программный модуль?

Программные модули в конфигурации системы 1С:Предприятие не являются самостоятельными программами в общепринятом понимании этого слова, поскольку они являются только частью всей *конфигурации задачи*. Программный модуль — это своего рода «контейнер» для размещения текстов процедур и функций, вызываемых системой во время исполнения задачи в определенные моменты работы. Поэтому программный модуль не имеет формальных границ своего описания типа: «Начало модуля» — «Конец модуля».

Место размещения конкретного программного модуля (тот самый «контейнер») предоставляется конфигуратором в тех точках конфигурации задачи, которые требуют описания специфических алгоритмов функционирования. Эти алгоритмы следует оформлять в виде процедур или функций, которые будут вызваны самой системой в заранее предусмотренных ситуациях (например, при нажатии кнопки в диалоговом окне).

Каждый отдельный программный модуль воспринимается системой как единое целое, поэтому все процедуры и функции программного модуля выполняются в едином контексте.

Контекст выполнения программного модуля

Каждый программный модуль связан с остальной частью конфигурации задачи. Эта связь называется контекстом выполнения модуля. Следует различать два вида контекста:

- глобальный контекст задачи;
- локальный контекст выполнения конкретного модуля.

Глобальный контекст образуется:

- значениями системных атрибутов, системными процедурами и функциями;
- значениями заданных в конфигураторе *констант, перечислений, регистров, видов расчета, групп видов расчета*;
- переменными, процедурами и функциями глобального программного модуля, объявленными с ключевым словом `Экспорт`.

Глобальный контекст виден всем программным модулям и определяет общую языковую среду конфигурации.

Локальный контекст модуля образуется тем конкретным местом конфигурации задачи, для которого использован программный модуль. Локальный контекст виден только конкретному программному модулю и определяет для модуля набор непосредственно доступных модулю значений агрегатных типов данных, их атрибутов и методов (см. «Виды программных модулей»). Однако, контекст модуля можно передать как объект в виде параметра при вызове процедур и функций (см. «Передача локального контекста программного модуля в качестве параметра»). Кроме того, контекст модуля определяет тот набор методов, которые доступны только в данном контексте (см. «Атрибуты и методы контекста Модуля формы», «Методы контекста Модуля формы элемента справочника» и т. п.). Локальный контекст предназначен для того, чтобы дать возможность управлять частными аспектами поведения задачи, присущими данному модулю.

Виды программных модулей

В системе 1С:Предприятие существуют следующие виды программных модулей (места размещения программных модулей в конфигурации задачи), доступных в конфигураторе:

Размещение	Момент запуска	Контекст выполнения
Глобальный модуль		
Размещается в корневом разделе конфигурации: Метаданные.	Запускается при старте всей прикладной задачи.	Глобальный модуль определяет глобальный контекст всей задачи.
Модуль Формы списка справочника		
Размещается в разделе конфигурации: Метаданные — Справочник — Форма списка.	Запускается при вызове формы списка справочника.	В модуле доступны: глобальный контекст, контекст *Модуля формы списка справочника*, в котором непосредственно доступен выбранный в списке элемент справочника и реквизиты формы списка справочника.

Модуль Формы группы справочника

Размещается в разделе конфигурации: Метаданные — Справочник — Форма группы.	Запускается при открытии формы группы справочника.	В модуле доступны: глобальный контекст, контекст *Модуля формы группы справочника*, в котором непосредственно доступны реквизиты текущей группы справочника и реквизиты формы.

Модуль Формы элемента справочника

Размещается в разделе конфигурации: Метаданные — Справочник — Форма элемента.	Запускается при открытии формы элемента справочника.	В модуле доступны: глобальный контекст, контекст *Модуля формы элемента справочника*, в котором непосредственно доступны реквизиты текущего элемента справочника и реквизиты формы.

Модуль Формы документа

Размещается в разделе конфигурации: Метаданные — Документ — Форма.	Запускается при открытии формы документа.	В модуле доступны: глобальный контекст, контекст *Модуля формы документа*, в котором непосредственно доступны реквизиты текущего документа и реквизиты формы документа.

Модуль документа

Размещается в разделе конфигурации: Метаданные — Документ — Модуль документа.	Запускается при проведении документа, при удалении проведенного документа, при снятии проведения, при выполнении архивации записей журнала расчетов, порожденных документом.	В модуле доступны: глобальный контекст, контекст *Модуля документа*, в котором непосредственно доступны реквизиты текущего документа.

Модуль Формы журнала документов

Размещается в разделе конфигурации: Метаданные — Журнал — Форма.	Запускается при вызове формы журнала документов.	В модуле доступны: глобальный контекст, контекст *Модуля формы журнала документов*, в котором непосредственно доступен выбранный в журнале документ и реквизиты формы журнала.

Модуль Формы журнала расчетов

Размещается в разделе конфигурации: Метаданные — Журнал расчетов — Форма.	Запускается при вызове формы журнала расчетов.	В модуле доступны: глобальный контекст, контекст *Модуля формы журнала расчетов*, в котором непосредственно доступны реквизиты журнала расчетов и реквизиты формы.

Модуль Формы списка счетов

Размещается в разделе конфигурации: Метаданные — План счетов.	Запускается при вызове формы списка счетов.	В модуле доступны: глобальный контекст, контекст *Модуля формы списка счетов*, в котором непосредственно доступен выбранный в списке счет и реквизиты формы списка счетов.

Модуль Формы счета

Размещается в разделе конфигурации: Метаданные — Справочник — счет.	Запускается при открытии формы счета.	В модуле доступны: глобальный контекст, контекст *Модуля формы счета*, в котором непосредственно доступны реквизиты текущего счета и реквизиты формы.

Модуль Формы журнала операций

Размещается в разделе конфигурации: Метаданные — Журнал операций — Форма.	Запускается при вызове формы журнала операций.	В модуле доступны: глобальный контекст, контекст *Модуля формы журнала операций*, в котором непосредственно доступны реквизиты журнала операций и реквизиты формы.

Модуль Формы операции

Размещается в разделе конфигурации: Метаданные — Операция.	Запускается при открытии формы операции.	В модуле доступны: глобальный контекст, контекст *Модуля формы операции*, в котором непосредственно доступны реквизиты текущей операции и реквизиты формы операции.

Модуль Формы журнала проводок

Размещается в разделе конфигурации: Метаданные — Журнал проводок — Форма.	Запускается при вызове формы журнала проводок.	В модуле доступны: глобальный контекст, контекст *Модуля формы журнала проводок*, в котором непосредственно доступны реквизиты журнала проводок и реквизиты формы.

Модуль Формы отчета

Размещается в разделе конфигуратора: Метаданные — Отчет — Форма.	Запускается при открытии диалоговой формы подготовки отчета.	В модуле доступны: глобальный контекст, контекст *Модуля формы отчета*, в котором непосредственно доступны реквизиты формы.

Модуль Формы обработки

Размещается в разделе конфигуратора: Метаданные — Обработка — Форма.	Запускается при открытии диалоговой формы обработки.	В модуле доступны: глобальный контекст, контекст *Модуля формы обработки*, в котором непосредственно доступны реквизиты формы.

Модуль вида расчета

Размещается в разделе конфигуратора: Метаданные — Вид расчета — Модуль вида расчета.	Запускается при расчете соответствующих записей журнала расчетов.	В модуле доступны: глобальный контекст, контекст *Модуля вида расчета*, в котором доступны реквизиты журнала расчетов.

Формат программного модуля

Исходный текст программного модуля может состоять из операторов и комментариев.

Комментарии

Комментарий используется для размещения в исходном тексте программного модуля всякого рода пояснений к работе модуля. Хорошим тоном программирования считается, когда исходный текст содержит исчерпывающий комментарий с описанием алгоритма. В режиме исполнения программы комментарии пропускаются. В тексте программного модуля комментарий начинается парой символов "//" и заканчивается концом строки. Это значит, что комментарий можно начинать с начала строки или записывать его после оператора на той же строке. После начала комментария писать оператор на той же строке нельзя, необходимо закончить комментарий концом строки.

Пример:

```
A=B; // Это - комментарий
// Это тоже комментарий
```

Формат операторов

Операторы имеют вид стандартного обращения к процедуре, за исключением оператора присваивания (A=B;) и управляющих конструкций (таких как `Для, Пока, Если`). Между собой операторы обязательно следует разделять символом ";" (точкой с запятой). Конец строки не является признаком конца оператора, т. е. операторы могут свободно переходить через строки и продолжаться на другой строке. Можно располагать произвольное число операторов на одной строке, разделяя их символом ";".

Операторы языка в программном модуле можно подразделить на две категории: операторы объявления переменных и исполняемые операторы.

Операторы объявления переменных создают имена переменных, которыми манипулируют исполняемые операторы.

Любой исполняемый оператор может иметь метку, используемую в качестве точки перехода в операторе Перейти.

В общем случае формат оператора языка следующий:

```
~метка:Оператор[(параметры)] [ДобКлючевоеСлово];
```

В качестве меток используются специальные идентификаторы, начинающиеся с символа '~' (тильда) и состоящие из последовательности букв, цифр и символов '_'. Чтобы пометить оператор, надо поместить перед ним метку и следующий за ней символ ":".

Пример:

```
~метка:А=В;
```

Имена переменных, процедур и функций

Именем переменной, объявленной процедуры или функции может быть любая последовательность букв, цифр и знаков подчеркивания "_", начинающаяся с буквы или знака подчеркивания "_". Вновь создаваемые имена не должны совпадать с зарезервированными словами языка или именами существующих процедур и функций, доступных на момент выполнения. Распознавание имен переменных, процедур и функций ведется без учета регистра букв.

Зарезервированные слова

Приведенные далее ***ключевые слова*** являются зарезервированными и не могут использоваться в качестве создаваемых имен переменных и объявляемых процедур и функций. В данном варианте языка каждое из ключевых слов имеет два представления — русское и английское. Английское представление является традиционным для языков программирования. Ключевые слова в русском и английском представлении могут свободно смешиваться в одном исходном тексте. Регистр букв ключевых слов не имеет значения. Ниже приведен список ключевых слов в обоих вариантах представления.

Если	If	Не	Not	Дата	Date
Тогда	Then	Знач	Val	Формат	Format
ИначеЕсли	Elsif	СтрДлина	StrLen	Разм	Dim
Иначе	Else	СокрЛ	TrimL	Вопрос	Do Query Box
КонецЕсли	Endlf	СокрП	TrimR	Контекст	Context
Цикл	Do	Лев	Left	Перем	Var
Для	For	Прав	Right	Перейти	Goto
По	To	Сред	Mid	Возврат	Return
Пока	While	Цел	Int	Продолжить	Continue
Функция	Function	Окр	Round	Прервать	Break
КонецПроцедуры	EndProcedure	Число	Number	И	And
КонецФункции	EndFunction	Строка	String	Или	Or
Предупреждение	DoMessageBox	КонецЦикла	EndDo	Процедура	Procedure

Структура программного модуля

Структуру программного модуля можно подразделить на следующие разделы:

- раздел определения переменных;
- раздел процедур и функций;
- раздел основной программы.

В конкретном программном модуле любой из разделов может отсутствовать.

Раздел определения переменных размещается от начала текста модуля до первого оператора Процедура или оператора Функция или любого исполняемого оператора. В этом разделе могут находиться только операторы объявления переменных Перем.

Раздел процедур и функций размещается от первого оператора Процедура или оператора Функция до любого исполняемого оператора вне тела описания процедур или функций.

Раздел основной программы размещается от первого исполняемого оператора вне тела процедур или функций до конца модуля. В этом разделе могут находиться только исполняемые операторы. Раздел основной программы исполняется в момент запуска модуля на выполнение (см. «Виды программных модулей»). Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо провести до первого вызова любой из процедур или функций модуля.

Специальные символы, используемые в исходном тексте

//	Двумя знаками «косая черта» начинается комментарий. Комментарием считается весь текст от знака "//" до конца текущей строки.
|	Вертикальная черта в начале строки используется только в строковых константах и означает, что данная строка является продолжением предыдущей (перенос строки), (см. «Строковые константы»).
~	Знаком тильда начинается метка оператора.
:	Двоеточием заканчивается метка оператора.

;	Точка с запятой является символом разделения операторов.
( )	В круглые скобки заключается список параметров методов, процедур и функций.
[ ]	В квадратные скобки заключается размерность массивов.
,	Запятая разделяет параметры в списке параметров методов, процедур и функций.
" "	В двойные кавычки заключаются строковые константы.
' '	В одинарные кавычки заключаются константы даты.
.	Десятичная точка в числовых константах. Разделитель, используемый в описаниях агрегатных типов данных.
+	Символ «плюс» обозначает операцию сложения.
-	Символ «минус» обозначает операцию вычитания.
*	Символ «звездочка» обозначает операцию умножения.
/	Символ «косая черта» обозначает операцию деления.
>	Правая угловая скобка обозначает логическую операцию «больше».
>=	Логическая операция «больше или равно».
<	Левая угловая скобка обозначает логическую операцию «меньше».
<=	Логическая операция «меньше или равно».
=	Знак равенства обозначает присвоение или логическую операцию «равно».
<>	Две угловые скобки обозначают логическую операцию «не равно».

Процедуры и функции программного модуля

Процедура

Секция описания процедуры.

Синтаксис:

```
Процедура <Имя_проц>([[Знач] <Парам1>[=<ДефЗнач>],
                ... , [Знач] <ПарамN>[=<ДефЗнач>]]) [Экспорт]
    //Объявления локальных переменных;
    //Операторы;
    ...
    [Возврат;]
    //Операторы;
    ...
КонецПроцедуры
```

Англоязычный Синтаксис:

```
Procedure <Имя_проц>([[Val] <Парам1>[=<ДефЗнач>],
                ... , [Val] <ПарамN>[=<ДефЗнач>]]) [Export]
    //Объявления локальных переменных;
    //Операторы;
    ...
    [Return;]
    //Операторы;
    ...
EndProcedure
```

Параметры:

<Имя_проц>	Назначает имя процедуры.
Знач	Необязательное ключевое слово, которое указывает на то, что следующий за ним параметр передается по значению, т. е. изменение значения формального параметра при выполнении процедуры никак не повлияет на фактический параметр, переданный при вызове процедуры. Если это ключевое слово не указано, то параметр процедуры передается по ссылке, то есть изменение внутри процедуры значения формального параметра приведет к изменению значения соответствующего фактического параметра.
<Парам1>, ..., <ПарамN>	Необязательный список формальных параметров, разделяемых запятыми. Значения формальных параметров должны соответсвовать значениям передаваемых при вызове процедуры фактических параметров. В этом списке определяются имена каждого из параметров так, как они используются в тексте процедуры. Список формальных параметров может быть пуст. **См. также:** «Передача параметров».
=<ДефЗнач>	Необязательная установка значения параметра по умолчанию. Параметры с установленными значениями по умолчанию можно располагать в любом месте списка формальных параметров. Если параметр при вызове процедуры опущен, то он принимает либо установленное по умолчанию значение (если оно есть) либо принимает «пустое» значение (значение неопреде-

	ленного типа). Если параметру не задано значение по умолчанию и он является последним в списке передаваемых параметров, то при вызове процедуры его нельзя опускать. Если параметру задано значение по умолчанию и он является последним в списке, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров и не ставить запятую перед опущенным параметром. Если параметру не задано значения по умолчанию, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров, но разделительную запятую надо ставить.
`Экспорт`	Необязательное ключевое слово, которое указывает на то, что данная процедура является доступной из других программных модулей. Имеет смысл только в *глобальном программном модуле*.
`//Объявления локальных переменных`	Объявляются локальные переменные, на которые можно ссылаться только в рамках этой процедуры (см. оператор `Перем`).
`//Операторы`	Исполняемые операторы процедуры.
`Возврат`	Необязательное ключевое слово, которое завершает выполнение процедуры и осуществляет возврат в точку программы, из которой было обращение к процедуре. Использование данного оператора в процедуре не обязательно.
`КонецПроцедуры`	Обязательное ключевое слово, обозначающее конец исходного текста процедуры, завершение выполнения процедуры. Возврат в точку, из которой было обращение к процедуре.

Описание:

Ключевое слово `Процедура` начинает секцию исходного текста, выполнение которого можно инициировать из любой точки программного модуля, просто указав `Имя_процедуры` со списком параметров (если параметры не передаются, то круглые скобки, тем не менее, обязательны). Если в *глобальном программном модуле* в теле описания функции использовано ключевое слово `Экспорт`, то это означает, что данная процедура является доступной из всех других программных модулей конфигурации. Ключевое слово `Экспорт` имеет смысл использовать только в *глобальном программном модуле*.

При выполнении оператора `Возврат` процедура заканчивается и возвращает управление в точку вызова. Если в тексте процедуры не встретился оператор `Возврат`, то после выполнения последнего исполняемого оператора происходит выполнение неявного оператора `Возврат`. Конец программной секции процедуры определяется по оператору `КонецПроцедуры`.

Переменные, объявленные в теле процедуры в разделе `Объявления_локальных_переменных`, являются локальными переменными данной процедуры, поэтому доступны только в этой процедуре (за исключением случая передачи их как параметров при вызове других процедур, функций или методов).

Замечание: ключевые слова Процедура, КонецПроцедуры, являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой (это может приводить к ошибкам выполнения модуля).

См. также: «Передача параметров»

Пример:

```
Перем Глоб;
// Описание процедуры
Процедура МояПроцедура(Пар1, Пар2, Пар3) Экспорт
    Глоб = Глоб + Пар1 + Пар2 + Пар3;
    Возврат;
КонецПроцедуры
Глоб = 123;
МояПроцедура(5, 6, 7);      // Вызов процедуры
```

Функция

Секция описания функции.

Синтаксис:

```
Функция <Имя_функции>([[Знач] <Парам1>[=<ДефЗнач>], ...
                , [Знач] <ПарамN> [=<ДефЗнач>]])[Экспорт]
    //Объявления локальных переменных;
    // Операторы;
    ...
    Возврат <ВозвращаемоеЗначение>;
    // Операторы;
```

```
    ...
КонецФункции
```

Англоязычный Синтаксис:

```
Function <Имя_функции> ([[Val] <Парам1>[=<ДефЗнач>], ...
                , [Val] <ПарамN> [=<ДефЗнач>]])[Export]
    //Объявления локальных переменных;
    // Операторы;
    ...
    Return <ВозвращаемоеЗначение>;
    // Операторы;
    ...
EndFunction
```

Параметры:

<Имя_функции>	Назначает имя функции.
Знач	Необязательное ключевое слово, которое указывает на то, что следующий за ним параметр передается по значению, т. е. изменение значения формального параметра при выполнении функции никак не повлияет на значение фактического параметра, переданного при вызове функции. Если это ключевое слово не указано, то параметр функции передается по ссылке, то есть изменение внутри функции значения формального параметра приведет к изменению значения соответствующего фактического параметра.
<Парам1>, ..., <ПарамN>	Необязательный список формальных параметров, разделяемых запятыми. Значения формальных параметров должны соответсвовать значениям передаваемых при вызове функции фактических параметров. В этом списке определяются имена каждого из параметров так, как они используются в тексте функции. Список формальных параметров может быть пуст. **См. также:** «Передача параметров».
=<ДефЗнач>	Необязательная установка значения параметра по умолчанию. Параметры с установленными значениями по умолчанию можно располагать в любом месте списка формальных параметров. Если параметр при вызове процедуры опущен, то он принимает либо установленное по умолчанию значение (если оно есть) либо принимает «пустое» значение (значение неопределенного типа). Если параметру не задано значение по умолчанию и он является последним в списке передаваемых параметров, то при вызове процедуры его нельзя опускать. Если параметру задано значение по умолчанию и он является последним в списке, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров и не ставить запятую перед опущенным параметром. Если параметру не задано значения по умолчанию, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров, но разделительную запятую надо ставить.
Экспорт	Необязательное ключевое слово, которое указывает на то, что данная функция является доступной из других программных модулей. Данное ключевое слово имеет смысл использовать только в *глобальном программном модуле*.
//Объявления локальных переменных	Объявляются локальные переменные, на которые можно ссылаться только в рамках этой функции (см. оператор Перем).
//Операторы	Исполняемые операторы функции.
Возврат	Ключевое слово, которое завершает выполнение функции и возвращает ВозвращаемоеЗначение в выражение, в котором используется функция. Использование данного ключевого слова в функции обязательно.
<ВозвращаемоеЗначение>	Выражение, значение которого содержит результат обращения к функции.
КонецФункции	Обязательное ключевое слово, обозначающее конец исходного текста функции.

Описание:

Ключевое слово Функция начинает секцию исходного текста функции, выполнение которой можно инициировать из любой точки программного модуля, просто указав Имя_функции со списком параметров (если параметры не передаются, то круглые скобки, тем не менее, обязательны). Если в *глобальном программном модуле* в теле описания функции использовано ключевое слово Экспорт, то это означает, что данная функция является доступной из всех других программных модулей конфигурации. Ключевое слово Экспорт имеет смысл использовать только в *глобальном программном модуле*.

Выполнение функции заканчивается обязательным оператором Возврат. Функции отличаются от процедур только тем, что возвращают Воз вращаемое Значение. Конец программной секции функции определяется по оператору КонецФункции.

Вызов любой функции в тексте программного модуля можно записывать как вызов процедуры, т. е. в языке допускается не принимать от функции возвращаемое значение.

Переменные, объявленные в теле функции в разделе Объявления_локальных_переменных, являются локальными переменными данной функции, поэтому доступны только в этой функции (за исключением случая передачи их как параметров при вызове других процедур, функций или методов).

Замечание: ключевые слова Функция, КонецФункции являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой (это может приводить к ошибкам выполнения модуля).

См. также: «Передача параметров»
Пример:

```
Перем Глоб;
// Описание функции
Функция МояФункция(Парам1, Парам2, Парам3) Экспорт
    Лок = Глоб + Парам1 + Парам2 + Парам3;
    Возврат Лок;
КонецФункции
Глоб = 123;
Рез = МояФункция(5, 6, 7);  // Вызов функции
```

Предварительное описание процедур и функций

В тексте программного модуля допускается предварительное описание процедур и функций без их определения.
Пример:

- фрагмента текста модуля, содержащий предварительное описание процедуры.

```
Процедура А(Парам1, Парам2) Далее
Процедура Б()
   Перем АА, ББ;
   ...
   А(АА, ББ);
   ...
КонецПроцедуры
Процедура А(Парам1, Парам2)
   ...
КонецПроцедуры
```

В приведенном фрагменте видно, что обращение к процедуре А (из процедуры Б) предшествует по тексту ее фактическому определению, но перед процедурой Б имеется предварительное описание процедуры А.

```
Процедура А(Парам1, Парам2) Далее
```

На то, что это предварительное описание указывает наличие ключевого слова Далее (англоязычный синоним — Forward), которое замещает в случае предварительного описания тело процедуры и ключевое слово КонецПроцедуры. Предварительное описание процедуры/функции может содержаться в любом месте текста модуля, где допускается фактическое определение процедуры или функции, а сам заголовок процедуры/функции должен в точности соответствовать заголовку в фактическом определении, включая наличие, если необходимо, ключевого слова Экспорт и имен формальных параметров.

Передача параметров

По умолчанию параметры методов, процедур и функций передаются по ссылке, то есть изменение внутри процедуры или функции значения формального параметра ведет к изменению значения соответствующего фактического параметра. При передаче параметра по значению изменение значения формального параметра никак не влияет на фактический параметр вызова процедуры. Для указания того, что тот или иной параметр следует передавать по значению, следует в исходном тексте процедуры или функции перед именем параметра записать ключевое слово Знач (англоязычный синоним — Val).

Если параметру задано значение по умолчанию и он является последним в списке, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров и не ставить запятую перед опущенным параметром.

Если параметру не задано значения по умолчанию, то при вызове процедуры его можно опускать в списке передаваемых фактических параметров, но разделительную запятую надо ставить.

Если параметр при вызове процедуры опущен, то он принимает либо установленное по умолчанию значение (если оно есть) либо принимает «пустое» значение (значение неопределенного типа).

Если при вызове метода, процедуры или функции параметры не передаются (пустой список параметров), то, тем не менее, круглые скобки обязательно требуется ставить.

Пример:

```
Перем Глоб;
// Описание функции
Функция МояФункция(Знач Пар1, Пар2, Пар3) Экспорт
   Лок = Глоб + Пар1 + Пар2 + Пар3;
   Пар1 = 40;
   Возврат Лок;
КонецФункции
Глоб = 123;
А = 10;
Рез = МояФункция(А, 6, 7); // Вызов функции
// Здесь Рез = 146, а переменная А = 10, несмотря на то, что в теле
// функции значение параметра Пар1 изменено на 40
```

Передача локального контекста программного модуля в качестве параметра

В языке есть возможность передавать локальный контекст программного модуля (см. «Контекст выполнения программного модуля», «Виды программных модулей») в качестве параметра процедуры или функции. Для этого в качестве фактического параметра при вызове процедуры или функции используется ключевое слово `Контекст` (англоязычный синоним — `Context`). Данная возможность позволяет, например, написать в глобальном программном модуле процедуры и функции (общие для многих модулей), которые, при их вызове из других модулей, будут исполняться с использованием конкретного локального контекста вызвавшего модуля.

Пример:

- Допустим, у нас есть несколько различных документов, причем у всех есть реквизит `"Склад"`. В каждом программном модуле формы редактирования документа нужна процедура выбора склада. В глобальном программном модуле напишем процедуру:

```
Процедура УстСклада(Конт) Экспорт
   // создаем объект типа "справочник"
   Скл = СоздатьОбъект("Справочник.Склады");
   // вызываем диалог для выбора элемента Справочника
   Скл.Выбрать("Выберите Склад", "");
   Если Скл.Выбран() = 1 Тогда
      // если в диалоге элемент Справочника был выбран,
      // то присваиваем его значение реквизиту Документа,
      // который доступен по локальному контексту
      Конт.Склад = Скл.ТекущийЭлемент();
   КонецЕсли;
КонецПроцедуры
```

- Здесь переменная Конт является формальным параметром, которому при вызове процедуры будет присвоен локальный контекст. В данном примере обращение к реквизитам и методам локального контекста происходит «через точку» после идентификатора Конт, например:

```
Конт.Склад = Скл.ТекущийЭлемент();
```

- Теперь в любом программном модуле конфигурации (в данном примере в любом Модуле формы документа) для выбора склада можно вызвать процедуру, передав ей свой локальный контекст в качестве параметра:

```
УстСклад(Контекст);
```

Пример исходного текста программного модуля

```
Перем ФлагПроводки; // Флаг проведения Документа
Перем ФлагИзменения; // Флаг изменения Документа
//———————————————————————————————————————
Функция КонтрольОстатка()
   // Функция возвращает текущее значение остатка товара на складе
   Ост = Регистр.ОстаткиТоваров.Остаток(Склад, Товар, "ОстатокТовара");
   Возврат Ост;
КонецФункции
//———————————————————————————————————————
Процедура УстФлагИзм()
```

```
   ФлагИзменения = 1;
КонецПроцедуры
//————————————————————————————————————————
Процедура ВводНового()
   // предопределенная процедура, вызываемая при вводе нового документа
   // установим дату Документа
   ДатаДок = РабочаяДата();
   // установим реквизиты Документа (по умолчанию)
   ТипНакладной = Перечисление.ТипыНакладных.Перемещение;
   Фирма = Константа.ДефФирма;
   Склад = Константа.ДефСклад;
   СкладПолучатель = Константа.ДефСклад;
КонецПроцедуры
//————————————————————————————————————————
Процедура ВыходноеСообщение()
   // процедура, которую мы будем вызывать при нажатии кнопки "ОК"
   Если (ФлагПроводки = 1) И (ФлагИзменения = 1) Тогда
      Предупреждение("Проведенный Документ был изменен! При
            | сохранении обязательно проведите Документ снова!");
   КонецЕсли;
КонецПроцедуры
//————————————————————————————————————————
// При входе в Форму запомним промежуточные переменные
ФлагПроводки = Проведен();
ФлагИзменения = 0;
```

Глава 2
Типы данных

Базовые типы данных

В языке поддерживаются следующие базовые типы данных:

- числовой;
- строковый;
- дата.

Числовым типом может быть представлено любое десятичное число. Над данными числового типа определены основные арифметические операции.

Строковым типом может задаваться любая последовательность символов, в том числе и пустая.

Типом ***дата*** может представляться любая корректная дата.

Правила преобразования типов данных

В процессе работы модуля или вычисления выражений может возникнуть необходимость в преобразовании типов данных отдельных значений. Для базовых типов определены следующие правила преобразования:

Число -> Строка

Если число не имеет форматных установок, то его строковым представлением является полное представление числа в формате с фиксированной точкой.

Дата -> Строка

Дата преобразованная к строковому типу имеет вид ДД.ММ.ГГ, где ГГ — две цифры года, ММ — числовое обозначение месяца (01, ..., 12), ДД — день месяца.

Строка -> Число

Строка преобразуется в число пока это возможно. Получившееся число считается результатом преобразования. (Например, строка "1.22 Glass" будет преобразована к числу 1.22). Если в начале строки не имеется ничего, что могло бы быть проинтерпретировано как число, то результат равен 0.

Дата -> Число

Результатом данного преобразования является численное представление даты.

Строка -> Дата

Если в начале строки содержится что-то, что может быть проинтерпретировано как строковое представление даты в виде ДД.ММ.ГГ, где ГГ — две цифры года, ММ — числовое обозначение месяца (01, ..., 12), ДД — день месяца, то будет произведено соответствующее преобразование. В противном случае значение даты будет нулевым.

Число -> Дата

Делается попытка взять целую часть Числа и проинтерпретировать как численное представление даты. Если число отрицательно, то итогом является нулевая дата.

Агрегатные типы данных

Агрегатные типы данных — это специализированные типы данных системы 1С:Предприятие, суть которых — отражение объектов предметной области и способ работы с ними.

Работа с агрегатными типами данных имеет существенные отличия от работы с обычными базовыми типами данных. Конкретные значения существующих агрегатных типов данных могут быть определены в программном модуле двумя способами:

- некоторые значения заранее известны в программном модуле из его глобального и локального контекста;
- другие значения могут быть определены с помощью системной функции `СоздатьОбъект`, которой в качестве параметра передается строка с именем агрегатного типа данных, созданного в конфигураторе.

Каждый агрегатный тип данных, как правило, имеет набор атрибутов и методов.

Атрибуты по свойствам напоминают переменные, т. е. им можно присваивать или читать их значения.

Методы — это те действия, которые может выполнять агрегатный тип данных. Методы могут иметь или не иметь возвращаемое значение. Если метод имеет возвращаемое значение, то он может размещаться в правой части оператора присваивания, в выражениях, в описании фактических параметров других вызываемых методов, процедур или функций.

Замечание. В тексте программного модуля вызов любого метода можно записывать отдельным оператором программы (как вызов процедуры) даже в том случае, если метод имеет возвращаемое значение. Другими словами, в языке допускается игнорировать возвращаемое значение.

Замечание. Работа с агрегатными типами данных языка напоминает работу с объектными типами в Visual Basic. Атрибуты агрегатного типа данных — аналогичны свойствам объектного типа в Visual Basic. Методы — аналогичны методам объектного типа в VisualBasic.

Замечание. При конфигурировании агрегатных типов данных, например Справочников, Документов, Регистров и т. п. не рекомендуется использовать в качестве идентификаторов создаваемых рекизитов существующие термины названий атрибутов и методов объектов.

Типичная последовательность работы с объектом агрегатного типа данных выглядит следующим образом:

- с помощью функции `СоздатьОбъект` создается объект агрегатного типа данных, и какой-либо переменной присваивается ссылка на него;
- объект позиционируется на нужном элементе данных;
- производятся различные манипуляции с объектом агрегатного типа данных через вызовы методов и обращения к его атрибутам.
- в случае, если объект агрегатного типа данных больше не нужен, он может быть отсоединен от переменной посредством переприсваивания переменной какого-либо значения базового типа (например, числа 0). Отсоединение объекта выполнять не обязательно.

Пример:

```
// Создаем объект типа "справочник.Сотрудники"
Сотр = СоздатьОбъект("Справочник.Сотрудники");
// Позиционируем созданный объект Сотр по известному наименованию
Сотр.НайтиПоНаименованию("Петров");
Если Сотр.Выбран() = 1 Тогда
   // если сотрудник найден, отобразим его оклад
   Предупреждение("Оклады" + Сотр.Оклад);
Иначе
   Предупреждение("Петров не найден" + " !!!");
КонецЕсли;
// отсоединяем объект
Сотр=0;
```

Замечание. Объект, созданный при помощи функции. `СоздатьОбъект`, изначально не определен, т. е. не содержит никакого конкретного значения. Чтобы начать с ним работать, его предварительно надо позиционировать (установить на конкретный *документ* или элемент *справочника*). Для документов позиционирование выполняется при помощи методов `НайтиДокумент`, `НайтиПоНомеру`, `ПолучитьДокумент`. Для справочников позиционирование выполняется при помощи методов `НайтиЭлемент`, `НайтиПоКоду`, `ПолучитьЭлемент`.

Замечание. Следует понимать, что в случае, если значение переменной, содержащей ссылку на объект агрегатного типа данных, присвоить другой переменной, то это не означает создание нового объекта, просто другая переменная будет содержать ссылку на тот же объект. Новые объекты создаются только с помощью обращения к функции `СоздатьОбъект`. Удаление объекта производится самой системой после того, когда не остается ни одной переменной, содержащей ссылку на объект.

Агрегатными типами данных называются следующие типы данных:

Константа — средство работы с постоянными (или условно постоянными) значениями. В *константах* хранится информация, которая не изменяется или изменяется достаточно редко. Например, название организации, почтовый адрес. Перечень *констант*, доступный в конкретной конфигурации, их названия и тип определяются в конфигураторе.

Справочник — средство для ведения списков однородных элементов данных. Помимо наименования элементов данных, списки могут содержать различную дополнительную информацию. Физическим аналогом *справочника* может являться картотека. Каждая карточка — это элемент *справочника*, а сведения, заносимые в карточку, являются реквизитами *справочника*. Перечень *справочников*, доступных в конкретной конфигурации, их названия и реквизиты определяются в конфигураторе.

Перечисление — средство работы с элементами данных, список возможных значений которых жестко задан (например, для *перечисления* «`ФормаОплаты`» можно задать возможные значения: «`Нал`», «`Безнал`»). В отличие от *справочников*, списки значений в *перечислении* задаются в процессе их создания в конфигураторе и при выполнении задачи не могут быть изменены. Состав *перечислений*, доступных в конкретной конфигурации, их названия и допустимые значения определяются в конфигураторе.

Документ — средство для ввода первичной информации о совершаемых хозяйственных операциях. Перечень *документов*, доступных в конкретной конфигурации, их названия, реквизиты и другие свойства определяются в конфигураторе.

Регистр — средство обработки и накопления сводной информации оперативного учета. Перечень *регистров*, доступных в конкретной конфигурации, их названия, измерения, ресурсы и другие свойства определяются в конфигураторе. (*Регистры доступны только при наличии компоненты «Оперативный учет»*)

ПланСчетов является служебным типом данных. Он предназначен для идентификации Плана счетов, созданного в метаданных. В основном он используется для передачи в качестве параметра различным процедурам и функциям компоненты «Бухгалтерский учет» и для выбора плана счетов в формах. Тип значения «ПланСчетов» не поддерживает никаких

данных в информационной базе, а список возможных значений этого типа данных определен планами счетов созданными в конфигурации. Значения типа «ПланСчетов» могут выступать как реквизиты диалога формы, как реквизиты документов, справочников и т. д. Для получения значения такого типа данных используется глобальный атрибут `ПланыСчетов`, который имеет в свою очередь набор атрибутов типа «ПланСчетов» соответсвующих имеющимся в конфигурации планам счетов. Кроме того, глобальный атрибут `ПланыСчетов` имеет методы для обхода всех существующих планов счетов. (*Данные типа ПланСчетов доступны только при наличии компоненты «Бухгалтерский учет»*)

Счет — это агрегатный тип данных для доступа к объектам данных — бухгалтерским счетам. Бухгалтерские счета используются компонентой «Бухгалтерский учет» системы 1С:Предприятие для идентификации разрезов синтетического учета наличия и движения средств. В общем виде смысл типа данных «Счет» вполне соответвует общепринятому понятию «Счет» в бухгалтерском учете. В конфигурации системы может быть создано несколько планов счетов. План счетов является фактически видом для значений типа «Счет». Например, реквизит типа «Счет» некоторого диалога может иметь конкретный вид (относиться к конкретному плану счетов) или быть неопределенного вида — то есть принимать значение различных планов счетов. (*Данные типа Счет доступны только при наличии компоненты. «Бухгалтерский учет»*)

ВидСубконто является служебным типом данных. Он предназначен для идентификации Вида субконто, созданного в метаданных. В основном он используется для передачи в качестве параметра различным процедурам и функциям компоненты «Бухгалтерский учет» и для выбора вида субконто в формах. Список возможных значений этого типа данных определен видами субконто, созданными в конфигурации. Значения типа «ВидСубконто» могут выступать как реквизиты диалога формы, как реквизиты документов, справочников и т. д. Для получения значения такого типа используется глобальный атрибут `ВидыСубконто`, который имеет в свою очередь набор атрибутов типа «ВидСубконто» соответствующих имеющимся видам субконто. Кроме того, глобальный атрибут `ВидыСубконто` имеет методы для обхода всех существующих видов субконто. (*Данные типа ВидСубконто доступны только при наличии компоненты «Бухгалтерский учет»*)

Операция — средство для манипулирования из встроенного языка данными бухгалтерских операций и проводок формируемых документом. Так как проводки в системе 1С:Предприятие принадлежат операциям, то управление и операциями и проводками выполняется объектом «Операция». (*Данные типа Операция доступны только при наличии компоненты «Бухгалтерский учет»*)

БухгалтерскиеИтоги — средство для организации доступа к бухгалтерским итогам в различных разрезах, за различные периоды и с разной степенью детализации. При наличии в системе 1С:Предприятие компоненты «Бухгалтерский учет» система автоматически реализует специальный механизм работы с бухгалтерскими итогами. Данный механизм обеспечивает хранение, динамический пересчет бухгалтерских итогов и их извлечение средствами встроенного языка. Система хранения бухгалтерских итогов поддерживается системой 1С:Предприятие автоматически на основе существующих планов счетов. При редактировании планов счетов — в конфигураторе или при работе с системой 1С:Предприятие — для счета могут быть установлены ряд свойств, которые влияют на организацию хранения бухгалтерских итогов: это признаки ведения валютного и количественного учета, а также включение аналитического учета по субконто. Изменение бухгалтерских итогов может производиться только проводками бухгалтерских операций. (*Данные типа БухгалтерскиеИтоги доступны только при наличии компоненты «Бухгалтерский учет»*)

ЖурналРасчетов — средство для учета расчетных действий по тем или иным объектам расчета. Каждая *строка журнала* расчетов соответствует одному расчетному действию — акту расчета, выполняемому по определенному алгоритму (*виду расчета*), в определенном временном интервале и имеющему результат. (*ЖурналыРасчетов доступны только при наличии компоненты «Расчет»*)

ВидРасчета — средство для выполнения расчетов по определенным алгоритмам через *журнал расчетов*. Перечень *видов расчета*, доступных в конкретной конфигурации, их названия и другие свойства определяются в конфигураторе. (*ВидыРасчета доступны только при наличии компоненты «Расчет»*)

ГруппаРасчетов — тип данных, предназначенный для объединения *видов расчета* по тому или иному признаку. Перечень *групп расчетов*, доступных в конкретной конфигурации, их названия и другие свойства определяются в конфигураторе. (*ГруппыРасчетов доступны только при наличии компоненты «Расчет»*)

Календарь — средство для ведения учета времени по календарным датам. Перечень *календарей*, доступных в конкретной конфигурации, их названия и другие свойства определяются в конфигураторе. (*Календари доступны только при наличии компоненты «Расчет»*)

Запрос — средство для выполнения обращения к документам, регистрам, документам, справочникам и журналам расчетов с целью получения сводной информации при формировании выходных отчетов. В программных модулях допускается создавать произвольное число объектов типа *Запрос* при помощи вызова системной функции `СоздатьОбъект`.

Текст — средство работы с текстовыми документами. В программных модулях допускается создавать произвольное число объектов типа *Текст* при помощи вызова системной функции `СоздатьОбъект`.

Таблица — средство работы с таблицами (отчетами). В программных модулях допускается создавать произвольное число объектов типа *Таблица*, при помощи вызова системной функции `СоздатьОбъект`.

СписокЗначений — средство для создания списка значений каких-либо данных и возможности в дальнейшем сортировать и выбирать нужные значения из списка. При добавлении в диалоговых формах полей типа «Список» или «Поле со списком», система автоматически создает объекты *СписокЗначений*, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создавать произвольное число объектов типа *СписокЗначений* при помощи вызова системной функции `СоздатьОбъект`.

ТаблицаЗначений — средство для создания списка значений каких-либо данных и возможности в дальнейшем сортировать и выбирать нужные значения из списка. При добавлении в диалоговых формах полей типа «Список» или «Поле со списком», система автоматически создает объекты *СписокЗначений*, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создавать произвольное число объектов типа *СписокЗначений* при помощи вызова системной функции `СоздатьОбъект`.

Картинка — средство для работы с графическими файлами. При добавлении в диалоговых формах и в таблицах полей типа «Картинка», система автоматически создает объекты *Картинка*, доступ к которым в языке возможен по идентификатору поля. В программных модулях допускается создавать произвольное число объектов типа *Картинка* при помощи вызова системной функции `СоздатьОбъект`.

Периодический — средство для работы с периодическими реквизитами справочников и периодическими константами. В программных модулях допускается создавать произвольное число объектов типа *Периодический* при помощи вызова системной функции `СоздатьОбъект`.

ФС — средство для работы с дисковыми файлами непосредственно из встроенного языка системы 1С:Предприятие. В программных модулях допускается создавать произвольное число объектов типа *ФС* при помощи вызова системной функции `СоздатьОбъект`. Кроме того, в глобальном контексте по умолчанию существует один уже созданный объект этого типа с именем ФС (имя объекта совпадает с названием агрегатного типа данных).

XBase — средство для работы с файлами баз данных DBF формата непосредственно из встроенного языка системы 1С:Предприятие. В программных модулях допускается создавать произвольное число объектов типа *XBase* при помощи вызова системной функции `СоздатьОбъект`.

Англоязычные синонимы названий агрегатных типов данных

Названия агрегатных типов данных	Англоязычные синонимы
Константа	Const
Справочник	Reference
Перечисление	Enum
Документ	Document
Регистр	Register
ПланСчетов	ChartOfAccounts
Счет	Account
ВидСубконто	SubcontoKind
Операция	Operation
БухгалтерскиеИтоги	BookkeepingTotals
ЖурналРасчетов	CalcJournal
ВидРасчета	CalculationKind
ГруппаРасчетов	CalculationGroup
Календарь	Calendar
Запрос	Query
Текст	Text
Таблица	Table
СписокЗначений	ValueList
ТаблицаЗначений	ValueTable
Картинка	Picture
Периодический	Perlodic
ФС	FS
XBase	Xbase

Атрибуты агрегатных типов данных

Атрибут — свойство агрегатного типа данных. В общем случае атрибуты могут размещаться в правой и в левой части оператора присваивания, в выражениях, быть параметром вызываемых методов, процедур или функций. Имена атрибутов можно использовать для того, чтобы определить или задать текущее значение атрибута конкретного объекта агрегатного типа данных. Некоторые атрибуты доступны только для чтения, это специально указывается при описании этого атрибута в данном руководстве. Атрибуты только для чтения не могут стоять в левой части оператора присваивания.

Доступ к атрибутам конкретного объекта агрегатного типа данных зависит от контекста программного модуля.

Если объект агрегатного типа данных входит (согласно контекста) в набор непосредственно доступных модулю значений агрегатных типов данных, то доступ к атрибуту — просто имя этого атрибута.

В других случаях, доступ к атрибутам конкретного объекта агрегатного типа данных происходит при помощи создания ссылки на объект. Объект агрегатного типа данных создается при помощи функции `СоздатьОбъект`. Для доступа к атрибуту объекта, имя атрибута пишется через точку после имени переменной, содержащей ссылку на этот объект.

Пример:

```
*
Спр = СоздатьОбъект("Справочник.Сотрудники");
А = Спр.Выбрать("Выберите объект", 0);
//Выбираем наименование и оклад как атрибуты справочника
Сообщить("Выбран сотрудник " + Спр.Наименование);
Сообщить("Оклад - " + Спр.Оклад );
*
Спр = СоздатьОбъект("Справочник.Сотрудники");
А = Спр.Выбрать("Выберите объект", 0);
Док = СоздатьОбъект("Документ.ПриказНаДоплату");
Док.Новый();
// Устанавливаем атрибут НомерДок
Док.НомерДок = "777";
// Устанавливаем атрибут ДатаДок
Док.ДатаДок = '14.04.96';
Док.ТипРасчета = ВидРасчета.ДоплатаСуммой;
```

Методы агрегатных типов данных

Методы — это те действия, которые может выполнять агрегатный тип данных. Методы могут иметь или не иметь возвращаемое значение. В тексте программного модуля вызов любого метода можно записывать отдельным оператором программы (как вызов процедуры), т. е. в языке допускается не принимать от методов возвращаемое значение. Если метод имеет возвращаемое значение, то он может размещаться в правой части оператора присваивания, в выражениях, в качестве фактических параметров других вызываемых методов, процедур или функций.

Синтаксис вызова методов конкретного агрегатного типа данных зависит от контекста программного модуля.

Если объект агрегатного типа данных входит (согласно контекста) в набор непосредственно доступных модулю значений агрегатных типов данных, то вызов метода — просто имя этого метода с указанием необходимых параметров.

В других случаях вызов метода конкретного агрегатного типа данных происходит при помощи создания ссылки на объект. Объект агрегатного типа данных создается при помощи функции СоздатьОбъект. Чтобы вызвать метод объекта, имя этого метода (с указанием необходимых параметров) пишется через точку после имени созданного объекта.

См. также: «Передача параметров»

Пример:

```
Спр = СоздатьОбъект("Справочник.Товары");
А = Спр.Выбрать("Выберите товар", 0);
Доку = СоздатьОбъект("Документ.НаклПрих");
Доку.Новый();                 // Вызываем метод
Доку.Склад = 22;
Доку.НомерДок = "777";        // Устанавливаем номер документа
Доку.ДатаДок = '14.04.96';    // Устанавливаем дату документа
Доку.АвтоВремяОтключить();    // Вызываем метод
Доку.УстановитьВремя(17, 30, 00);
Доку.НоваяСтрока();           // Вызываем метод
Доку.Товар = Спр.ТекущийЭлемент();
Доку.Количество = 100;
```

Глава 3
Объявление переменных

В языке переменные не обязательно объявлять в явном виде. Неявным определением переменной является первое ее появление в левой части оператора присваивания. Возможно также явное объявление переменной при помощи оператора `Перем`. Тип переменной определяется типом присвоенного ей значения. Не допускается использование в выражениях переменных с неопределенным значением (то есть переменных, которым никогда не присваивалось значения).

Переменные могут объединяться в массивы. В данной реализации программы предусмотрены только одномерные массивы.

Оператор объявления переменной

Перем

Объявление переменной в явном виде.

Синтаксис:

```
Перем <Имя_переменной> [[<Размерность>]] [Экспорт];
```

Англоязычный синоним:

```
Var:
```

Параметры

`<Имя_переменной>`	Имя переменной.
`[<Размерность>]`	Необязательная числовая константа (записывается в квадратных скобках), указывается только в том случае, если объявляется массив и должна представляться в виде положительного целого числа.
`Экспорт`	Необязательное ключевое слово `Экспорт` указывает, что данная переменная станет доступна для других модулей. Использование данного ключевого слова имеет смысл только в *глобальном программном модуле*.

Описание:

Оператор `Перем` в явном виде объявляет переменную.

Пример:

```
Перем Очень_Нужная_Переменная;
Перем Массив_Для_Хранения[10];
```

Область использования переменной

Область использования переменных зависит от места их определения в конфигурации задачи. Существует три области, в которых можно объявить переменные:

- В разделе определения переменных глобального программного модуля. Если переменные определены с ключевым словом `Экспорт` (см. оператор `Перем`), то это будут глобальные переменные.
- В разделе определения переменных модуля. Это переменные модуля.
- В процедуре или функции. Это локальные переменные.

Глобальные переменные доступны для использования в исполняемых операторах, выражениях, в любой процедуре и функции любого программного модуля конфигурации задачи.

Переменные модуля доступны для использования в исполняемых операторах, выражениях, в любой процедуре и функции того программного модуля, в пределах которого они объявлены.

Локальные переменные доступны в пределах той процедуры или функции, в которой они объявлены.

Если переменная определена как глобальная переменная, то она видна из всех процедур и функций любого программного модуля конфигурации задачи. Если же переменная определена внутри процедуры, то её областью видимости является данная процедура или функция. Таким образом, если две переменные с одинаковыми именами используются в двух различных процедурах модуля, и имя этой переменной не упоминается как глобальная переменная, то это две различные переменные, локальные для процедур. Если же переменная определена как глобальная переменная, то любое использование имени этой переменной будет приводить к обращению к одной и той же переменной.

Единственный способ создать для процедуры локальную переменную с именем, совпадающим с именем переменной, определенной как глобальная переменная — это объявить ее явно при помощи оператора `Перем`.

Глава 4
Выражения и оператор присваивания

Выражения

Выражение — это математическая или логическая формула, по которой вычисляется значение. Математическое выражение может стоять справа от знака равенства в операторах присваивания, быть параметром процедур или функций, индексом массива переменных. Логическое выражение может быть условием в управляющих операторах `Если`, `Пока`, `Для`. Выражения состоят из констант, переменных и функций, связанных символами логических и/или арифметических операций.

Арифметические операции

В языке определены следующие виды арифметических операций:

сложение	(Оп1 + Оп2)
вычитание	(Оп1 - Оп2)
умножение	(Оп1 * Оп2)
деление	(Оп1 / Оп2)
остаток от деления	(Оп1 % Оп2)
унарный минус	(-Оп1)

Арифметические операции имеют один или два операнда, в зависимости от типа которых операция имеет ту или иную семантику. Тот или иной семантический вариант операции определяется по первому операнду. В случае несовпадения типа второго операнда с требуемым, значение преобразуется к требуемому типу в соответствии с правилами преобразования типов. Если тип первого операнда не соответствует ни одному из допустимых типов, то в зависимости от ситуации может производиться преобразование типов или возбуждаться состояние ошибки выполнения.

Сложение определено для следующих типов операндов:	Число + Число
	Дата + Число (к дате прибавляется число дней)
Вычитание определено для следующих типов операндов:	Число – Число
	Дата – Число (от даты отнимается число дней)
	Дата – Дата (результатом является число дней между датами)
Умножение:	Число * Число
Деление:	Число / Число
Остаток от деления:	Число % Число

Замечание. При выполнении операции % (остаток от деления) оба операнда операции округляются до целого значения.

Операция конкатенации

Операция конкатенации ("+") используется для того, чтобы присоединить одну строку к другой. Длина результирующей строки равна сумме длин соединяемых строк. В случае несовпадения типа данных второго или последующих операндов со строковым типом, их значение преобразуется к строковому типу в соответствии с правилами преобразования типов.

Пример:

- Для того, чтобы удалить ненужные пробелы, используются системные функции `СокрЛ` и `СокрП`.

```
ФИО = СокрП(Фамилия) + " " + СокрП(Имя) + " " + СокрП(Отчество);
```

Логические операции

Логическая операция сравнивает операнды и вырабатывает логическое значение: «истина» или «ложь». Существует два вида логических операций: операции сравнения и булевы операции. В операциях сравнения сравниваются два значения. Булевы операции выполняются над логическими значениями, реализуя булеву алгебру. Символы булевых операций могут комбинироваться, образуя составные операции.

Операции сравнения:

В языке определены следующие виды операций сравнения:

больше	(Оп1 > Оп2)
больше или равно	(Оп1 >= Оп2)
равно	(Оп1 = Оп2)
не равно	(Оп1 <> Оп2)

меньше	(Оп1 < Оп2)
меньше или равно	(Оп1 <= Оп2)

Операции сравнения определены для следующих типов операндов:

Больше	Число > Число
	Строка > Строка
	Дата > Дата
Больше или равно:	Число >= Число
	Строка >= Строка
	Дата >= Дата
Меньше:	Число < Число
	Строка < Строка
	Дата < Дата
Меньше или равно:	Число <= Число
	Строка <= Строка
	Дата <= Дата
Равно:	Число = Число
	Строка = Строка
	Дата = Дата
	АгрегатныйТип = АгрегатныйТип
Не равно:	Число <> Число
	Строка <> Строка
	Дата <> Дата
	АгрегатныйТип <> АгрегатныйТип

Булевы операции:

В языке определены следующие виды булевых операций:

И или AND	конъюнкция (булево И)
ИЛИ или OR	дизъюнкция (булево ИЛИ)
НЕ или NOT	логическое отрицание (булево отрицание НЕ)

Числовые константы

Константа числового типа представляется в виде:
['+'|'-']{'0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'}[.{'0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'}]

Пример:

```
0   123   -15   +24.11      0.245
```

Константы даты

Дата задается в формате 'ДД.ММ.ГГ' или 'ДД.ММ.ГГГГ' (дата заключается в одиночные кавычки). Неопределенная дата задается как '00.00.00'.

Пример:

```
'12.04.95'      '01.01.1996'
```

Строковые константы

Константа строкового типа представляется как любая последовательность символов, заключенных в двойные кавычки.

Пример:

```
"Ура заработала"
"Это самая правильная строка"
" "
```

Кроме того, допускаются «многострочные» строковые константы. В исходном тексте многострочные константы могут задаваться двумя способами:

- Первый вариант: В данном случае между фрагментами, представляющими отдельные строки многострочной константы, не должно встречаться никаких символов, за исключением пробелов, переводов строки и строк комментариев.

Пример:

```
Стр = "Первая строка"    // пример строковой константы
"Вторая строка"
"Третья строка";
```

- Второй вариант: В данном примере значение константы полностью идентично предыдущему примеру. Отличие заключается в том, что каждая отдельная составляющая не замыкается кавычками, а на каждой последующей строке помещен символ переноса строки "|". В этом варианте комментарии между открывающей и закрывающей кавычками не допускаются.

Пример:

```
Стр = "Первая строка
   |Вторая строка
   |Третья строка";
```

Строковые выражения

Строковые выражения могут использоваться в качестве параметров методов, процедур и функций, либо в правой части оператора присваивания. Строковым выражением может быть отдельная строка или сложное сочетание строковых констант, функций и операций. То, что выражение является строковым, определяется по первому операнду. В случае несовпадения типа второго и последующих операндов со строковым типом, их значения преобразуются к требуемому типу в соответствии с правилами преобразования типов. Чтобы указать, что выражение является строковым, можно, например, начинать его с пустой строки символов "".

Пример:

```
Стр = СокрП("Фамилия" + ", " + Имя + ", " + " отчество");
Стр1 = "" + Докум.Цена + Валюта.СокрНаим;
```

Логические выражения

В логических выражениях происходит вычисление логического значения (истина/ложь). Обычно логические выражения используются в управляющих операторах, в которых на основании результата логического выражения определяется дальнейшая последовательность выполнения программы. Логические выражения вычисляются слева направо. Для того, чтобы избежать неоднозначности и управлять последовательностью операндов, следует применять круглые скобки.

Уровни старшинства логических операций:

Уровень 1	операнды, заключенные в скобки
Уровень 2	NOT
Уровень 3	AND
Уровень 4	OR

Пример:

```
Стр = "Угадал!";
а = ВвестиСтроку(Стр, "Назови пароль!", 15);
Если (а=1) И ((Стр = "пароль") ИЛИ (Стр = "Пароль")) Тогда
   Сообщить(Стр);
КонецЕсли;
```

Оператор присваивания

Синтаксис:

```
<Назначение> = <Источник>;
```

Параметры:

<Назначение>	В качестве <Назначения> может выступать простая переменная, элемент массива, атрибут агрегатного типа данных (который допускает запись). Элемент массива задается как Имя_массива[Индексное_выражение].
<Источник>	Выражение.

Описание:

Оператор присваивания (символ "=") означает присваивание значения <Источник> переменной, обозначенной как <Назначение>.

Пример:

```
А = В;
Стр1 = "777";
ДатаДок = '14.04.96';
```

Глава 5
Управляющие операторы

Управляющие конструкции

Если

Оператор условного выполнения.
Синтаксис:
```
Если <Логическое_выражение> Тогда
   //операторы;
[ИначеЕсли <Логическое_выражение> Тогда]
   //операторы;
[Иначе]
   //операторы;
КонецЕсли;
```
Англоязычный Синтаксис:
```
If <Логическое_выражение> Then
   //операторы;
[Elsif <Логическое_выражение> Then]
   //операторы;
[Else]
   //операторы;
Endlf;
```
Параметры:

<Логическое_выражение>	Логическое выражение.
Тогда	*Операторы* следующие за Тогда выполняются, если результатом *логического выражения* в конструкции Если является истина.
//операторы	Исполняемый оператор или последовательность таких операторов.
ИначеЕсли	Логическое выражение, следующее за ключевым словом ИначеЕсли вычисляется только тогда, когда условия в Если и всех предшествующих ИначеЕсли оказались ложными. Операторы следующие за конструкцией ИначеЕсли — Тогда выполняются, если результатом *логического выражения* в данном ИначеЕсли является истина.
Иначе	Операторы, следующие за ключевым словом Иначе выполняются, если результаты логических выражений в конструкции Если и всех предшествующих конструкциях ИначеЕсли оказались ложными.
КонецЕсли	Ключевое слово, которое завершает структуру оператора условного выполнения.

Описание:
Оператор Если управляет выполнением программы, основываясь на результате одного или более логических выражений. Оператор может содержать любое количество групп операторов, возглавляемых конструкциями
ИначеЕсли — Тогда.
Пример:
```
Если (РабочаяДата() > '01.01.95') И (Сег = "Понедельник") Тогда
   Сообщить("Сегодня" + " " + Сег + " " + РабочаяДата());
КонецЕсли;
```

Пока

Оператор выполнения цикла.
Синтаксис:
```
Пока <Логическое_выражение> Цикл
   //операторы;
КонецЦикла;
```
Англоязычный Синтаксис:
```
While <Логическое_выражение> Do
   //операторы;
EndDo;
```

Параметры:

<Логическое_выражение>	Логическое выражение.
Цикл	Операторы, следующие за ключевым словом Цикл, выполняются, пока результатом *логического выражения* является истина.
//операторы;	Исполняемый оператор или последовательность таких операторов.
КонецЦикла	Ключевое слово, которое завершает структуру оператора цикла.

Описание:

Оператор цикла Пока предназначен для циклического повторения операторов, находящиеся внутри конструкции Цикл ... КонецЦикла. Цикл выполняется пока *логическое выражение* — истина. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Пример:

```
Док = СоздатьОбъект("Документ.БригадныйНаряд");
// Будем просматривать документы в интервале дат
Док.ВыбратьДокументы('11.01.88', '11.01.99');
// Цикл по всем Документам
Пока (Док.ПолучитьДокумент() = 1) Цикл
   // отобразим Документ в строке состояния
   Состояние(Док.Вид() + " " + Док.НомерДок + " " + Док.ДатаДок);
   Док.ВыбратьСтроки();
   // вложенный цикл по всем строкам Документа
   Пока(Док.ПолучитьСтроку() > 0) Цикл
      Если Док.КодРабот = 104 Тогда
         Сообщить(" " + Док.НомерДок);
      КонецЕсли;
   КонецЦикла;
КонецЦикла;
```

Для

Оператор выполнения цикла.

Синтаксис:

```
Для <Имя_переменной> = <Выражение1> По <Выражение2> Цикл
   //Операторы;
КонецЦикла;
```

Англоязычный Синтаксис:

```
For <Имя_переменной> = <Выражение1> To <Выражение2> Do
   //Операторы;
EndDo;
```

Параметры:

<Имя_переменной>	Идентификатор переменной (счетчика цикла), значение которой автоматически увеличивается при каждом повторении цикла.
<Выражение 1>	Числовое выражение, которое задает начальное значение, присваиваемое счетчику цикла при первом проходе цикла.
По	Синтаксическая связка для параметра <Выражение2>.
<Выражение 2>	Максимальное значение счетчика цикла. Когда переменная Имя_переменной становится больше чем <Выражение2>, выполнение оператора цикла Для прекращается. Величина приращения счетчика при каждом выполнении цикла равна 1.
Цикл	Операторы следующие за ключевым словом Цикл выполняются, пока значение переменной Имя_переменной меньше значения <Выражение2>.
//Операторы	Исполняемый оператор или последовательность таких операторов.
КонецЦикла	Ключевое слово, которое завершает структуру оператора цикла.

Описание:

Оператор цикла Для предназначен для циклического повторения операторов, находящиеся внутри конструкции Цикл ... КонецЦикла. Перед началом выполнения цикла значение <Выражение1> присваивается переменной <Имя_переменной>. Значение <Имя_переменной> автоматически увеличивается при каждом проходе цикла. Цикл выполняется пока значение переменной <Имя_переменной> меньше или равно значению <Выражение2>. Величина приращения счетчика при каждом выполнении цикла равна 1. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Пример:

```
// Выведем все строки текста
```

```
Выв = СоздатьОбъект("Текст");
Выв.Открыть("ТТТ");
Для i = 1 по Выв.КоличествоСтрок() Цикл
   Сообщить(Выв.ПолучитьСтроку(i));
КонецЦикла;
```

Попытка

Обработка исключительных ситуаций.

Синтаксис:

```
Попытка
   // Некоторые действия;
Исключение
   // Обработка исключительной ситуации;
КонецПопытки;
```

Англоязычный Синтаксис:

```
Try
   // Некоторые действия;
Except
   // Обработка исключительной ситуации;
EndTry;
```

Параметры:

`// Некоторые действия`	Исполняемый оператор или последовательность таких операторов.
`// Обработка исключительной ситуации`	Исполняемый оператор или последовательность операторов, которые обрабатывают исключительную ситуацию.
`Исключение`	Операторы, следующие за ключевым словом Исключение выполняются, если при выполнении последовательности операторов `<// Некоторые действия>` произошла ошибка времени выполнения.
`КонецПопытки`	Ключевое слово, которое завершает структуру оператора обработки исключительных ситуаций.

Описание:

Оператор `Попытка` управляет выполнением программы, основываясь на возникающие при выполнении модуля ошибочные (исключительные) ситуации и определяет обработку этих ситуаций.

В качестве ошибочных (исключительных) ситуаций воспринимаются ошибки времени выполнения модуля. Не предусмотрено определяемых пользователем исключений.

Если при выполнении последовательности операторов <// Некоторые действия> произошла ошибка времени выполнения, то выполнение оператора, вызвавшего ошибку прерывается и управление передается на первый оператор последовательности операторов <// Обработка исключительной ситуации>. При этом управление будет передано даже в том случае, если ошибку вызвал оператор, находящийся в процедуре или функции, вызванной из <// Некоторые действия>. Если ошибка произошла в вызванной процедуре или функции, то ее выполнение будет прервано, а локальные переменные уничтожены. Это справедливо для любой вложенности вызовов. После выполнения последовательности операторов <// Обработка исключительной ситуации> управление передается на следующий за ключевым словом `КонецПопытки` оператор. Если же последовательность <// Некоторые действия> выполнилась без ошибок, то последовательность операторов <// Обработка исключительной ситуации> будет пропущена и управление также будет продолжено с оператора, следующего за ключевым словом `КонецПопытки`.

Конструкции `Попытка-Исключение-КонецПопытки` могут быть вложенными. При этом, при возникновении исключительной ситуации управление будет передано на самый «внутренний» обработчик, начинающийся с ключевого слова `Исключение`. Последовательность операторов <// Обработка исключительной ситуации> может содержать оператор `ВызватьИсключение` (англоязычный синоним Raise). Выполнение данного оператора прекращает выполнение последовательности <// Обработка исключительной ситуации> и производится поиск более «внешнего» обработчика. Если таковой есть, то управление передается на его первый оператор. Если нет, то выполнение модуля прекращается с выдачей сообщения о первоначально возникшей ошибке. Оператор `ВызватьИсключение` может встречаться только внутри операторных скобок `Исключение ... КонецПопытки`.

В выдаче диагностики помощь может оказать встроенная функция `ОписаниеОшибки()` (англоязычный синоним `GetErrorDescription`). Эта функция не имеет параметров, а в качестве значения возвращает описание ошибки, такое же, как было бы выдано в окне сообщений, в случае отсутствия обработчика исключительной ситуации, за исключением описания модуля и строки, в которой ошибка возникла. Применять данную функцию имеет смысл только при выполнении последовательности операторов <// Обработка исключительной ситуации>, так как в ином случае она вернет пустую строку.

Пример:

```
Процедура СформироватьВExcel()
   Попытка
```

```
      // пытаемся обратиться к программе MS Excel
      Табл = СоздатьОбъект("Excel.Application");
   Исключение
      Предупреждение(ОписаниеОшибки() +
           "; Excel не установлен на данном компьютере!");
      Возврат;
   КонецПопытки;
   // Формирование отчета
   ...
КонецПроцедуры
```

Управляющие операторы

Перейти

Безусловная передача управления на другой оператор программы.

Синтаксис:

```
Перейти <Метка>;
```

Англоязычный синоним:

```
Goto
```

Параметры:

<Метка>	Метка другого исполняемого оператора в программе.

Описание:

Безусловная передача управления на другой оператор программы. Оператор `Перейти` передает управление от одного оператора к другому. `<Метка>` в этом операторе не должна быть меткой перехода на оператор `Процедура` или `Функция`.

Область действия оператора `Перейти` ограничивается программным модулем, процедурой или функцией; он не может передать управление за пределы программного модуля, процедуры или функции.

Оператор безусловного перехода не может быть использован для передачи управления на операторы, находящиеся внутри конструкций: `Пока ... КонецЦикла`, `Для ... КонецЦикла`, `Если ... ИначеЕсли ... Иначе ... КонецЕсли`, `Попытка ... Исключение ... КонецПопытки` извне этих конструкций.

Пример:

```
   // Выведем строки текста
   Выв = СоздатьОбъект("Текст");
   Выв.Открыть("ТТТ");
   Для i = 1 По Выв.КоличествоСтрок() Цикл
      Если i = 10 Тогда
         Перейти ~М1;
      КонецЕсли;
      Сообщить(Выв.ПолучитьСтроку(i));
      Перейти ~М2;
~М1:  Сообщить("Это десятая строка");
~М2: КонецЦикла;
```

Продолжить

Передача управления в начало цикла.

Синтаксис:

```
Продолжить;
```

Англоязычный синоним:

```
Continue;
```

Описание:

Передает управление в начало цикла. Оператор `Продолжить` немедленно передает управление в начало цикла, где производится вычисление и проверка условий выполнения цикла. Оператор `Продолжить` может использоваться только внутри конструкций операторов цикла `Пока` и `Для`.

Пример:

```
// Выведем строки текста начиная с 5
Выв = СоздатьОбъект("Текст");
Выв.Открыть("ТТТ");
Для i = 1 По Выв.КоличествоСтрок() Цикл
   Если i < 5 Тогда
      Продолжить;
```

```
   КонецЕсли;
   Сообщить(Выв.ПолучитьСтроку(i));
КонецЦикла;
```

Прервать

Прерывает выполнение цикла.

Синтаксис:

```
Прервать
```

Англоязычный синоним:

```
Break
```

Описание:

Прерывает выполнение цикла. Оператор `Прервать` приводит к немедленному прекращению выполнения цикла и передает управление первому оператору, следующему за конструкцией оператора `Пока` или `Для`. Оператор `Прервать` может использоваться только внутри конструкций операторов цикла `Пока` и `Для`.

Пример:

```
// Выведем строки текста с 1 по 10
Выв = СоздатьОбъект("Текст");
Выв.Открыть("ТТТ");
Для i = 1 По Выв.КоличествоСтрок() Цикл
   Сообщить(Выв.ПолучитьСтроку(i));
   Если i=10 Тогда
      Прервать;
   КонецЕсли;
КонецЦикла;
```

Возврат

Завершение процедуры или функции.

Синтаксис:

```
Возврат [<Выражение>]
```

Англоязычный синоним:

```
Return
```

Параметры:

`<Выражение>`	Выражение, значение которого содержит результат обращения к функции. Этот параметр обязателен для оператора `Возврат` в теле описания функции, но не может использоваться в процедуре.

Описание:

Оператор `Возврат` завершает выполнение процедуры или функции и передает управление в вызвавшую точку программы. Оператор `Возврат` обязателен в функции и необязателен в процедуре. Если в процедуре не используется оператор `Возврат`, то его неявное выполнение происходит вслед за последним исполняемым оператором процедуры. Данный оператор может применяться только в теле описания процедуры или функции, причем `<Выражение>` может задаваться только в случае использования оператора в контексте функции.

Пример:

```
Функция ДеньНедели(Номер)
   Если Номер =1 Тогда
      Return "понедельник";
   ИначеЕсли Номер =2 Тогда
      Возврат "вторник";
   ИначеЕсли Номер =3 Тогда
      Возврат "среда";
   ИначеЕсли Номер =4 Тогда
      Возврат "четверг";
   ИначеЕсли Номер =5 Тогда
      Возврат "пятница";
   ИначеЕсли Номер =6 Тогда
      Возврат "суббота";
   ИначеЕсли Номер =7 Тогда
      Возврат "воскресенье";
   Иначе
      Возврат "ошибка";
   КонецЕсли;
КонецФункции
```

Специальные конструкции языка

#ЗагрузитьИзФайла

Переключение загрузки программного модуля на загрузку из текстового файла.

Синтаксис:

```
#ЗагрузитьИзФайла <ИмяФайла>
```

Англоязычный синоним:

```
ftLoadFromFile
```

Параметры:

<ИмяФайла>	Имя файла, содержащего исходный текст программного модуля (записывается без кавычек и скобок).

Описание:

Данная конструкция переключает загрузку программного модуля на загрузку из указанного файла. Специальная конструкция `#ЗагрузитьИзФайла` должна записываться в первой строке программного модуля с первой позиции. Ее использование рекомендуется для отладки, чтобы быстро отлаживать код какого-либо программного модуля без промежуточного сохранения всей конфигурации. Данная конструкция переключает загрузку программного модуля конфигурации на загрузку из указанного файла. Содержимое программного модуля конфигурации при этом игнорируется.

Для редактирования собственно файла, откуда загружается текст программного модуля, можно использовать встроенный текстовый редактор системы 1С:Предприятие , открывая его через главное меню «Файл» — «Открыть». Далее, если выбрать в главном меню «Действия» — «Текст модуля», то это позволит при редактировании использовать свойства контекстного выделения цветом синтаксических конструкций языка и установленную для программных модулей ширину табуляции.

Данную возможность можно использовать для отладки любых программных модулей. Считывание текста файла и его компиляция происходит в момент открытия окна формы (документа, отчета и т. д.). Таким образом, в режиме исполнения, после внесения изменений и записи файла, содержащего текст модуля, следует просто закрыть и открыть окно формы заново. Внесенные в текст файла изменения сразу сработают, что позволяет отлаживать систему без промежуточного сохранения всей конфигурации.

После окончания отладки текста программного модуля в файле, следует перенести текст из файла в программный модуль конфигурации.

Замечание: `#ЗагрузитьИзФайла` является не оператором, а специальной конструкцией, поэтому не должна заканчиваться точкой с запятой.

Пример:

```
#ЗагрузитьИзФайла NEW_MOD.TXT
```

Глава 6
Системные константы

Системные константы — это константы, которые доступны в любом программном модуле конфигурации. Они могут размещаться в правой части оператора присваивания, в выражениях, могут использоваться в качестве параметров вызываемых методов, процедур или функций. Имена системных констант можно использовать только для того, чтобы получить их значения. Системные константы не могут стоять в левой части оператора присваивания.

Строковые системные константы

РазделительСтраниц

Содержит строковое значение перевода страницы текста.

Синтаксис:

```
РазделительСтраниц
```

Англоязычный синоним:

```
PageBreak
```

Описание:

Системная константа `РазделительСтраниц` содержит строковое значение перевода страницы текста. Данная системная константа применяется для вставки или поиска в тексте строкового значения перевода страницы текста.

Пример:

```
Выв=СоздатьОбъект("Текст");
Выв.ДобавитьСтроку("Страница 1");
Выв.ДобавитьСтроку(РазделительСтраниц);
Выв.ДобавитьСтроку("Страница 2");
Выв.Показать();
```

РазделительСтрок

Содержит строковое значение перевода строки текста.

Синтаксис:

```
РазделительСтрок
```

Англоязычный синоним:

```
LineBreak
```

Описание:

Системная константа `РазделительСтрок` содержит строковое значение перевода строки текста. Данная системная константа применяется для вставки или поиска в тексте строкового значения перевода строки текста

Пример:

```
Выв=СоздатьОбъект("Текст");
Выв.ДобавитьСтроку("Страница 1");
Выв.ДобавитьСтроку(РазделительСтрок);
Выв.ДобавитьСтроку("Страница 2");
Выв.Показать();
```

СимволТабуляции

Содержит строковое значение символа табуляции.

Синтаксис:

```
СимволТабуляции
```

Англоязычный синоним:

```
TabSymbol
```

Описание:

Системная константа `СимволТабуляции` содержит строковое значение символа табуляции. Данная системная константа применяется для вставки или поиска в тексте строкового значения символа табуляции.

Пример:

```
Выв=СоздатьОбъект("Текст");
Выв.ДобавитьСтроку("Страница 1");
Выв.ДобавитьСтроку(СимволТабуляции);
Выв.ДобавитьСтроку("Страница 2");
Выв.Показать();
```

Глава 7
Системные процедуры и функции

Системные процедуры и функции доступны в любом программном модуле конфигурации. Доступ к системным процедурам и функциям в текстах программных модулей непосредственный, т. е. просто имя процедуры или функции (без предварительной ссылки на какой-либо объект).

Вызов любой функции в тексте программного модуля можно записывать как вызов процедуры, т. е. в языке допускается игнорировать возвращаемое значение.

Математические функции

Окр

Округлить число.

Синтаксис:

```
Окр(<Число1>, [<Число2>], [<Способ>])
```

Англоязычный синоним:

```
Round
```

Параметры:

<Число1>	Числовое выражение, значение которого надо округлить.
<Число2>	Необязательный параметр. Числовое выражение, значение которого — формат округления (число знаков дробной части (если <Число2> отрицательно, то округляется до соответствующего количества знаков целой части). Значение этого параметра по умолчанию — 0.
<Способ>	Необязательный параметр. Числовое выражение — способ округления: 0 — если при округлении 1.5 = 1; 1 — если при округлении 1.5 = 2. Значение по умолчанию — 0.

Возвращаемое значение:

Числовое значение результата округления.

Описание:

Функция Окр возвращает числовое значение результата округления <Число1> до <Число2> знаков дробной части (если <Число2> отрицательно, то округляется до соответствующего количества знаков целой части). Параметр <Число2> может быть опущен, при этом принимается, что <Число2> = 0.

Пример:

```
ОкруглЦена = Окр(ЦЕНА, -2);
```

Цел

Получить целую часть числа.

Синтаксис:

```
Цел(<Число>)
```

Англоязычный синоним:

```
Int
```

Параметры:

<Число>	Числовое выражение.

Возвращаемое значение:

Числовое значение целой части переданного в качестве параметра числа.

Описание:

Функция Цел возвращает целую часть переданного в качестве параметра числа, полностью отсекая дробную часть.

Пример:

```
МожноКупить = Цел(Наличность/Цена);
```

Мин

Определить минимальное значение.

Синтаксис:

```
Мин(<Элемент1>, ..., <ЭлементN>)
```

Англоязычный синоним:

```
Min
```

Параметры:

<Элемент1>, ..., <ЭлементN>	Список выражений базовых типов данных.

Возвращаемое значение:

Значение минимального элемента из списка <Элемент1>, ..., <ЭлементN>

Описание:

Функция Мин определяет минимальное значение из списка <Элемент1>, ..., <ЭлементN>.

Тот или иной семантический вариант функции определяется по типу данных первого параметра. В случае несовпадения типа второго и последующих параметров с требуемым, их значения преобразуются к требуемому типу в соответствии с правилами преобразования типов. Если тип первого операнда не соответствует ни одному из допустимых типов, то в зависимости от ситуации может производиться преобразование типов или возбуждаться состояние ошибки исполнения.

Пример:

```
МинимЦена = Мин(Цена1, Цена2, Цена3);
```

Макс

Определить максимальное значение.

Синтаксис:

```
Макс(<Элемент1>, ..., <ЭлементN>)
```

Англоязычный синоним:

```
Max
```

Параметры:

<Элемент1>, ..., <ЭлементN> Список выражений базовых типов данных.

Возвращаемое значение:

Значение максимального элемента из списка <Элемент1>, ..., <ЭлементN>

Описание:

Функция Макс определяет максимальное значение из списка <Элемент1>, ..., <ЭлементN>.

Тот или иной семантический вариант функции определяется по типу данных первого параметра. В случае несовпадения типа второго и последующих параметров с требуемым, их значения преобразуются к требуемому типу в соответствии с правилами преобразования типов. Если тип первого операнда не соответствует ни одному из допустимых типов, то в зависимости от ситуации может производиться преобразование типов или возбуждаться состояние ошибки исполнения.

Пример:

```
МаксимЦена = Макс(Цена1, Цена2, Цена3);
```

Лог10

Вычислить десятичный логарифм числа.

Синтаксис:

```
Лог10(<Число>)
```

Англоязычный синоним:

```
Log10
```

Параметры:

<Число> Числовое выражение.

Возвращаемое значение:

Числовое значение результата вычисления десятичного логарифма.

Описание:

Функция Лог10 вычисляет и возвращает десятичный логарифм числа. При отрицательном или нулевом значении параметра возвращаемое значение — 0.

Пример:

```
Шкала = Лог10(Частота);
```

Лог

Вычислить натуральный логарифм числа.

Синтаксис:

```
Лог(<Число>)
```

Англоязычный синоним:

```
Ln
```

Параметры:

<Число> Числовое выражение.

Возвращаемое значение:

Числовое значение результата вычисления натурального логарифма.

Описание:

Функция Лог вычисляет и возвращает натуральный логарифм числа. При отрицательном или нулевом значении параметра возвращаемое значение — 0.

Пример:

```
Шкала = Лог(Частота);
```

Строковые функции

СтрДлина

Получить длину строки.
Синтаксис:
СтрДлина(<Строка>)
Англоязычный синоним:
StrLen
Параметры:
<Строка> Строковое выражение.
Возвращаемое значение:
Числовое значение длины строки.
Описание:
Функция СтрДлина получает в качестве параметра строку и возвращает числовое значение ее длины.
Пример:
```
Длина = СтрДлина("Измерим длину строки");
```

ПустаяСтрока

Проверить строку на наличие значащих символов.
Синтаксис:
ПустаяСтрока(<Строка>)
Англоязычный синоним:
IsBlankString
Параметры:
<Строка> Строковое выражение.
Возвращаемое значение:
Числовое значение. 1 — пустая строка или только пробелы. 0 — не пустая строка.
Описание:
Функция ПустаяСтрока проверяет переданный параметр <Строка> на наличие значащих символов (любых кроме пробелов) и возвращает число 1, если строка пустая или содержит только пробелы, 0 , если строка не пустая.
Пример:
```
Если ПустаяСтрока(Буфер) = 1 Тогда
   // если Буфер содержит только пробелы, то его удаляем
   Буфер = "";
КонецЕсли;
```

СокрЛ

Отбросить в строке стоящие слева пробелы.
Синтаксис:
СокрЛ(<Строка>)
Англоязычный синоним:
TrimL
Параметры:
<Строка> Строковое выражение.
Возвращаемое значение:
Строковое значение.
Описание:
Функция СокрЛ получает в качестве параметра строку, отсекает стоящие слева пробелы и возвращает результирующую строку.
Пример:
```
СтрЦена = СокрЛ(Цена) + " руб.";
```

СокрП

Отбросить в строке стоящие справа пробелы.
Синтаксис:
СокрП(<Строка>)
Англоязычный синоним:
TrimR
Параметры:
<Строка> Строковое выражение.

Возвращаемое значение:
Строковое значение.
Описание:
Функция СокрП получает в качестве параметра строку, отсекает стоящие справа пробелы и возвращает результирующую строку.
Пример:
```
ФИО = СокрП(Фамилия) + " " + СокрП(Имя) + " " + СокрП(Отчество);
```

СокрЛП

Отбросить в строке пробелы, стоящие слева и справа.
Синтаксис:
СокрЛП(<Строка>)
Англоязычный синоним:
TrimAll
Параметры:

<Строка>	Строковое выражение.

Возвращаемое значение:
Строковое значение.
Описание:
Функция СокрЛП получает в качестве параметра строку, отсекает стоящие слева и справа пробелы, и возвращает результирующую строку.
Пример:
```
ФИО = СокрЛП(Фамилия) + " " + СокрП(Имя) + " " + СокрП(Отчество);
```

Лев

Выбрать в строке символы слева.
Синтаксис:
Лев (<Строка>, <Число>)
Англоязычный синоним:
Left
Параметры:

<Строка>	Строковое выражение.
<Число>	Числовое выражение.

Возвращаемое значение:
Строковое значение.
Описание:
Функция Лев получает в качестве параметра строку, выбирает первые слева символы строки, числом равные <Число>, и возвращает в качестве значения результирующую строку.
Пример:
```
Фамилия = Лев(ФИО, 15);
```

Прав

Выбрать в строке символы справа.
Синтаксис:
Прав(<Строка>, <Число>)
Англоязычный синоним:
Right
Параметры:

<Строка>	Строковое выражение.
<Число>	Числовое выражение.

Возвращаемое значение:
Строковое значение.
Описание:
Функция Прав получает в качестве параметра строку, выбирает крайние справа символы строки, числом равные <Число>, и возвращает в качестве значения результирующую строку.
Пример:
```
Отчество = Прав(ФИО, 15);
```

Сред

Выбрать подстроку.

Синтаксис:
```
Сред(<Строка>, <Число1>, <Число2>)
```
Англоязычный синоним:
```
Mid
```
Параметры:

<Строка>	Строковое выражение.
<Число1>	Числовое выражение, начальный номер символа.
<Число2>	Числовое выражение, количество символов.

Возвращаемое значение:
Строковое значение.
Описание:
Функция Сред возвращает подстроку параметра <Строка>, начиная с символа с номером <Число1> общим количеством <Число2>. Позиции в строке считаются, начиная с 1. Параметр <Число2> может быть опущен, тогда выделяются все символы начиная с символа с номером <Число1> до конца строки.
Пример:
```
Имя = Сред(ФИО, 10, 12);
```

Найти

Найти вхождение подстроки.
Синтаксис:
```
Найти(<Строка1>, <Строка2>)
```
Англоязычный синоним:
```
Find
```
Параметры:

<Строка1>	Строковое выражение места поиска.
<Строка2>	Строковое выражение шаблона поиска.

Возвращаемое значение:
Числовое значение позиции первого знака найденной подстроки.
Описание:
Функция Найти находит вхождение строки <Строка2> как подстроки в <Строка1>. Возвращает числовое значение позиции первого знака найденной подстроки (первая позиция имеет индекс 1). Если не находит — возвращает число 0.
Пример:
```
Симв = Найти(ФИО, "Борис");
```

СтрЗаменить

Заменить вхождение подстроки другим значением.
Синтаксис:
```
СтрЗаменить(<СтрИсточник>, <СтрПодстрока>, <СтрЗамены>)
```
Англоязычный синоним:
```
StrReplace
```
Параметры:

<СтрИсточник>	Строковое выражение места поиска.
<СтрПодстрока>	Строковое выражение шаблона поиска.
<СтрЗамены>	Строковое выражение замены.

Возвращаемое значение:
Результирующая строка.
Описание:
Функция СтрЗаменить находит вхождение строки <СтрПодстрока> как подстроки в <СтрИсточник> и заменяет его на <СтрЗамены>. Результирующую строку возвращает в качестве собственного значения. Исходная строка не изменяется.
Пример:
```
Результат = СтрЗаменить(СтрИсточник, СтрПодстрока, СтрЗамены);
```

СтрЧислоВхождений

Вычислить число вхождений подстроки.
Синтаксис:
```
СтрЧислоВхождений(<СтрИсточник>, <СтрПодстрока>)
```
Англоязычный синоним:
```
StrCountOccur
```
Параметры:

<СтрИсточник> Строковое выражение места поиска.
<СтрПодстрока> Строковое выражение шаблона поиска.

Возвращаемое значение:
Число вхождений.
Описание:
Функция СтрЧислоВхождений вычисляет число вхождений строки <СтрПодстрока> как подстроки в <СтрИсточник>.
Пример:
```
Результат = СтрЧислоВхождений(СтрИсточник, СтрПодстрока);
```

СтрКоличествоСтрок

Определить число строк в многострочном тексте.
Синтаксис:
```
СтрКоличествоСтрок(<Строка>)
```
Англоязычный синоним:
```
StrLineCount
```
Параметры:
<Строка> Строковое выражение, в котором строки разделены символами перевода строк.
Возвращаемое значение:
Число строк в многострочном тексте.
Описание:
Функция СтрКоличествоСтрок вычисляет число строк в многострочном тексте (строки разделены символами перевода строк).
Пример:
```
Рез = СтрКоличествоСтрок(ДлТекст);
```
См. также: РазделительСтрок

СтрПолучитьСтроку

Получить строку из многострочного текста по номеру.
Синтаксис:
```
СтрПолучитьСтроку(<Текст>, <НомерСтроки>)
```
Англоязычный синоним:
```
StrGetLine
```
Параметры:
<Текст> Строковое выражение, в котором строки разделены символами перевода строк.
<НомерСтроки> Числовое выражение, начальный номер символа.
Возвращаемое значение:
Строковое значение.
Описание:
Функция СтрПолучитьСтроку возвращает строку с номером <НомерСтроки> из многострочного текста <Текст> (строки разделены символами перевода строк).
Пример:
```
ВтораяСтрока = СтрПолучитьСтроку(ДлТекст, 2);
```
См. также: РазделительСтрок

Врег

Преобразовать символы строки в верхний регистр.
Синтаксис:
```
ВРег(<Строка>)
```
Англоязычный синоним:
```
UpPeг
```
Параметры:
<Строка> Строковое выражение.
Возвращаемое значение:
Строковое значение.
Описание:
Функция ВРег преобразует все символы строки в верхний регистр. Результирующую строку возвращает в качестве собственного значения. Исходная строка не изменяется.
Пример:
```
Загл = ВРег("маленькие");
```

Нрег

Преобразовать символы строки в нижний регистр.

Синтаксис:

```
НРег(<Строка>)
```

Англоязычный синоним:

```
Lower
```

Параметры:

<Строка>	Строковое выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция НРег преобразует все символы строки в нижний регистр. Результирующую строку возвращает в качестве собственного значения. Исходная строка не изменяется.

Пример:

```
Загл = НРег("БОЛЬШИЕ");
```

OemToAnsi

Преобразовать строку в кодировку ANSI.

Синтаксис:

```
OemToAnsi(<Строка>)
```

Англоязычный синоним:

```
OemToAnsi
```

Параметры:

<Строка>	Строковое выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция OemToAnsi используется для перевода строки из DOS-кодировки в Windows-Кодировку. Результирующая строка возвращается в качестве собственного значения. Исходная строка не изменяется.

Пример:

```
Загл = OemToAnsi(ТекстDOS);
```

AnsiToOem

Преобразовать строку в кодировку OEM.

Синтаксис:

```
AnsiToOem(<Строка>)
```

Англоязычный синоним:

```
AnsiToOem
```

Параметры:

<Строка>	Строковое выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция AnsiToOem используется для перевода строки из Windows-кодировки в DOS-Кодировку. Результирующая строка возвращается в качестве собственного значения. Исходная строка не изменяется.

Пример:

```
Загл = AnsiToOem(ТекстWin);
```

Симв

Преобразовать код символа в символ.

Синтаксис:

```
Симв(<КодСимвола>)
```

Англоязычный синоним:

```
Chr
```

Параметры:

<КодСимвола>	Числовое выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция Chr преобразует код символа в символ. Возвращает результирующий символ в виде строкового значения.

Пример:

```
Буква_я = Симв(255);
```

КодСимв

Преобразовать символ в код символа.

Синтаксис:

```
КодСимв(<Символ>)
```

Англоязычный синоним:

```
Asc
```

Параметры:

<Символ>	Строковое выражение.

Возвращаемое значение:

Числовое значение.

Описание:

Функция КодСимв преобразует символ в код символа. Возвращает результирующий код символа в виде числового значения.

Пример:

```
КодБуквы_я = КодСимв("я");
```

Функции работы с датой

РабочаяДата

Установить/получить текущую рабочую дату.

Синтаксис:

```
РабочаяДата(<Дата>, <РежимСменыРабДаты>)
```

Англоязычный синоним:

```
WorkingDate
```

Параметры:

<Дата>	Необязательный параметр. Выражение типа «дата».
<РежимСменыРабДаты>	Необязательный параметр. Режим автоматической смены рабочей даты в полночь: 0 — не менять; 1 — менять с предупреждением; 2 — менять автоматически без предупреждения. Если параметр не указан, то режим не меняется и действует установка, выбранная в параметрах системы.

Возвращаемое значение:

Значение текущей рабочей даты (на момент до выполнения функции).

Описание:

При помощи функции РабочаяДата можно установить/получить значение рабочей даты, выбранной в текущем сеансе работы программы (которая может изменяться при помощи главного меню Сервис-Параметры-Общие-РабочаяДата).

Пример:

```
// оклад сотрудника на текущую рабочую дату
Сотрудник.Оклад.Получить(РабочаяДата());
```

ТекущаяДата

Возвратить текущую дату.

Синтаксис:

```
ТекущаяДата()
```

Англоязычный синоним:

```
CurDate
```

Возвращаемое значение:

Результирующая дата.

Описание:

Функция ТекущаяДата возвращает текущую (системную) дату. Возвращаемое значение — результирующая дата.

Пример:

```
ДатаТекущая = ТекущаяДата();
```

ДобавитьМесяц

Добавляет к указанной дате заданное число месяцев.

Синтаксис:

ДобавитьМесяц(<Дата>, <ЧислоМесяцев>)
Англоязычный синоним:
AddMonth
Параметры:

<Дата>	Выражение со значением типа «дата».
<ЧислоМесяцев>	Числовое выражение, задающее число месяцев, добавляемое к дате.

Возвращаемое значение:
Результирующая дата.
Описание:
Функция ДобавитьМесяц добавляет к указанной дате заданное число месяцев.
Пример:

```
ДатаЧерезТриМесяца = ДобавитьМесяц(РабочаяДата(), 3);
```

НачМесяца

Определить дату начала месяца.
Синтаксис:
НачМесяца(<Дата>)
Англоязычный синоним:
BegOfMonth
Параметры:

<Дата>	Выражение со значением типа «дата».

Возвращаемое значение:
Результирующая дата.
Описание:
Функция НачМесяца определяет дату начала месяца для указанной даты.
Пример:

```
ДатаНачалаМесяца = НачМесяца(РабочаяДата());
```

КонМесяца

Определить дату конца месяца.
Синтаксис:
КонМесяца(<Дата>)
Англоязычный синоним:
EndOfMonth
Параметры:

<Дата>	Выражение со значением типа «дата».

Возвращаемое значение:
Результирующая дата.
Описание:
Функция КонМесяца определяет дату конца месяца для указанной даты.
Пример:

```
ДатаКонцаМесяца = КонМесяца(РабочаяДата());
```

НачКвартала

Определить дату начала квартала.
Синтаксис:
НачКвартала(<Дата>)
Англоязычный синоним:
BegOfQuart
Параметры:

<Дата>	Выражение со значением типа «дата».

Возвращаемое значение:
Результирующая дата.
Описание:
Функция НачКвартала определяет дату начала квартала для указанной даты.
Пример:

```
ДатаНачалаКвартала = НачКвартала(РабочаяДата());
```

КонКвартала

Определить дату конца квартала.

Синтаксис:
КонКвартала(<Дата>)
Англоязычный синоним:
EndOfQuart
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Результирующая дата.
Описание:
Функция КонКвартала определяет дату конца квартала для указанной даты.
Пример:
```
ДатаКонцаКвартала = КонКвартала(РабочаяДата());
```

НачГода

Определить дату начала года.
Синтаксис:
НачГода(<Дата>)
Англоязычный синоним:
BegOfYear
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Результирующая дата.
Описание:
Функция НачГода определяет дату начала года для указанной даты.
Пример:
```
ДатаНачалаГода = НачГода(РабочаяДата());
```

КонГода

Определить дату конца года.
Синтаксис:
КонГода(<Дата>)
Англоязычный синоним:
EndOfYear
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Результирующая дата.
Описание:
Функция КонГода определяет дату конца года для указанной даты.
Пример:
```
ПатаКонцаГода = КонГода(РабочаяДата());
```

НачНедели

Определить дату начала недели.
Синтаксис:
НачНедели(<Дата>)
Англоязычный синоним:
BegOfWeek
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Результирующая дата.
Описание:
Функция НачНедели определяет дату начала недели для указанной даты.
Пример:
```
ДатаНачалаНедели = НачНедели(РабочаяДата());
```

КонНедели

Определить дату конца недели.

Синтаксис:
КонНедели(<Дата>)
Англоязычный синоним:
EndOfWeek
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Результирующая дата.
Описание:
Функция КонНедели определяет дату конца недели для указанной даты.
Пример:
```
ДатаКонцаНедели = КонНедели(РабочаяДата());
```

ДатаГод

Определить год указанной даты.
Синтаксис:
ДатаГод(<Дата>)
Англоязычный синоним:
GetYear
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением определенного года.
Описание:
Функция ДатаГод определяет год указанной даты.
Пример:
```
ЗначениеГода = ДатаГод(РабочаяДата());
```

ДатаМесяц

Определить месяц указанной даты.
Синтаксис:
ДатаМесяц(<Дата>)
Англоязычный синоним:
GetMonbh
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением номера месяца года.
Описание:
Функция ДатаМесяц определяет месяц указанной даты.
Пример:
```
ЗначениеМесяца = ДатаМесяц(РабочаяДата());
```

ДатаЧисло

Определить день месяца указанной даты.
Синтаксис:
ДатаЧисло(<Дата>)
Англоязычный синоним:
GetDay
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением определенного дня месяца.
Описание:
Функция ДатаЧисло определяет день месяца указанной даты.
Пример:
```
ЗначениеДняМесяца = ДатаЧисло(РабочаяДата());
```

НомерНеделиГода

Определить номер недели в году для указанной даты.

Синтаксис:
НомерНеделиГода(<Дата>)
Англоязычный синоним:
GetWeekOfYear
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением определенного номера недели в году.
Описание:
Функция НомерНеделиГода определяет номер недели в году для указанной даты.
Пример:
```
ЗначениеНедели = НомерНеделиГода(РабочаяДата());
```

НомерДняГода

Определить день года указанной даты.
Синтаксис:
НомерДняГода(<Дата>)
Англоязычный синоним:
GetDayOfYear
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением определенного дня года.
Описание:
Функция НомерДняГода определяет порядковый день в году для указанной даты.
Пример:
```
ЗначениеДняГода = НомерДняГода(РабочаяДата());
```

НомерДняНедели

Определить день недели указанной даты.
Синтаксис:
НомерДняНедели(<Дата>)
Англоязычный синоним:
GetDayOfWeek
Параметры:
<Дата> Выражение со значением типа «дата».
Возвращаемое значение:
Число со значением определенного порядкового дня недели.
Описание:
Функция НомерДняНедели определяет день недели указанной даты. Возвращаемое значение — число со значением определенного порядкового дня недели (1 — понедельник, 2 — вторник, ..., 7 — воскресенье).
Пример:
```
ЗначениеДняНедели = НомерДняНедели(РабочаяДата());
```

ПериодСтр

Строковое представление периода.
Синтаксис:
ПериодСтр(<ДатаНачалаПериода>, <ДатаКонцаПериода>)
Англоязычный синоним:
Period$tr
Параметры:
<ДатаНачалаПериода> Дата — начальная дата периода.
<ДатаКонцаПериода> Дата — конечная дата периода.
Возвращаемое значение:
Символьная строка — представление периода.
Описание:
Внутри системы все периоды задаются интервалом дат — дата начала и дата конца. Функция ПериодСтр предназначена для того, чтобы в случае, если период фактически является кварталом, месяцем, полугодием отобразить его соответствующим образом, например "1 квартал 1997 г".

Функция ПериодСтр возвращает информацию о периоде бухгалтерских итогов в виде строки символов. Строка может использоваться для выдачи ин-форации о периоде итогов в отчетах.

Пример:

```
Дата1='01.01.97';
Дата2='31.03.97';
Сообщить(ПериодСтр(Дата1, Дата2));  //Выводит "1 квартал 1997 г."
```

НачалоСтандартногоИнтервала

Устанавливает вариант задания начала стандартного интервала отображения журнала документов.

Синтаксис:

```
НачалоСтандартногоИнтервала(<Вариант>)
```

Англоязычный синоним:

```
BegOfStandrdRange
```

Параметры:

<Вариант>	Необязательный параметр. Выражение со значением типа строка или дата. Возможные строковые значения параметра: "День", "Месяц", "Квартал", "Год", "Day", "Month", "Quarter", "Year". Значением типа «дата» задается конкретная дата начала интервала журнала документов. Если параметр не задан, то установка сделанная в параметрах системы не меняется, а только возвращается текущий вариант.

Возвращаемое значение:

Возвращает текущий установленный вариант.

Описание:

Функция НачалоСтандартногоИнтервала устанавливает вариант зада ния начала стандартного интервала отображения журнала документов. Вариан может изменяться при помощи главного меню Сервис-Параметры-Общие.

Пример:

```
НачалоСтандартногоИнтервала("Месяц");
```

КонецСтандартногоИнтервала

Устанавливает вариант задания конца стандартного интервала ото бражения журнала документов.

Синтаксис:

```
КонецСтандартногоИнтервала(<Вариант>)
```

Англоязычный синоним:

```
EndOfStandardRange
```

Параметры:

<Вариант>	Необязательный параметр. Выражение со значением типа строка или дата. Возможные строковые значения параметра: "День", "Месяц", "Квартал", "Год", "Day", "Month", "Quarter", "Year". Значением типа «дата» задается конкретная дата конца интервала журнала документов. Если параметр не задан, то установка сделанная в параметрах системы не меняется, а только возвращается текущий вариант.

Возвращаемое значение:

Возвращает текущий установленный вариант.

Описание:

Функция КонецСтандартногоИнтервала устанавливает вариант задания конца стандартного интервала отображения журнала документов. Вариант может изменяться при помощи главного меню Сервис-Параметры-Общие.

Пример:

```
КонецСтандартногоИнтервала("Месяц");
```

Функции работы с временем

ТекущееВремя

Возвратить текущее время

Синтаксис:

```
ТекущееВремя(<Час>, <Мин>, <Сек>)
```

Англоязычный синоним:

```
CurrentTime
```

Параметры:

<Час>	Необязательный параметр. Имя переменной, куда система возвращает числовое значение часа текущего времени.
<Мин>	Необязательный параметр. Имя переменной, куда система возвращает числовое значение минут

текущего времени.

<Сек> Необязательный параметр. Имя переменной, куда система возвращает числовое значение секунд текущего времени.

Возвращаемое значение:

Текущее время в текстовом виде.

Описание:

Функция ТекущееВремя возвращает текущее (системное) время в текстовом виде. Кроме того, если в качестве параметров при вызове функции передать переменные, то функция вернет в них числовые значения текущего часа, минуты, секунды.

Пример:

```
Сообщить("Сейчас " + ТекущееВремя());
```

Функции преобразования типов

Дата

Преобразовать параметр в дату.

Синтаксис 1:

```
Дата(<Параметр>)
```

Синтаксис 2:

```
Дата(<Год>, <Месяц>, <Число>)
```

Англоязычный синоним:

```
Date
```

Параметры:

<Параметр> Числовое выражение.

<Год> Числовое выражение. Год указывается 4-х знач-ным числом (вместе с веком).

<Месяц> Числовое выражение.

<Число> Числовое выражение.

Возвращаемое значение:

Значение типа «дата».

Описание:

Функция Дата преобразует значение переданных параметров в дату, руководствуясь принятыми правилами преобразования типов и возвращает значение даты.

Пример:

```
// преобразование из строки
ДатаРожд = Дата('06.03.1958');
```

Строка

Преобразовать параметр в строку.

Синтаксис:

```
Строка(<Параметр>)
```

Англоязычный синоним:

```
String
```

Параметры:

<Параметр> Выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция Строка преобразует параметр в строку, руководствуясь принятыми правилами преобразования типов и возвращает значение строки.

```
Стр = Строка(ДатаРожд + 1);
```

Число

Преобразовать параметр в число.

Синтаксис:

```
Число(<Параметр>)
```

Англоязычный синоним:

```
Number
```

Параметры:

<Параметр> Выражение.

Возвращаемое значение:

Числовое значение.

Описание:

Функция Число преобразует параметр в число, руководствуясь принятыми правилами преобразования типов и возвращает значение числа.

Пример:

```
ВР = Число(Вар + Род);
```

Функции работы с позицией документа

СформироватьПозициюДокумента

Формирует позицию документа согласно переданным параметрам

Синтаксис 1:

```
СформироватьПозициюДокумента(<Докум>, <ФлагСмещения>)
```

Синтаксис 2:

```
СформироватьПозициюДокумента(<Дата>, <Час>, <Мин>, <Сек>, <ФлагКонцаСекунды>)
```

Англоязычный синоним:

```
MakeDocPosition
```

Параметры:

<Докум>	Значение документа (или его позиция), позиция которого определяется.
<ФлагСмещения>	Необязательный параметр. Число: -1 (минус единица) — предыдущая позиция. 1 — следующая позиция. 0 — текущая позиция. Значение по умолчанию — 0. Использование данного параметра позволяет получить предыдущую или последующую позицию от уже имеющейся позиции.
<Дата>	Значение типа «дата», на которую формируется позиция документа.
<Час>	Значение типа «число», которое определяет час, на который формируется позиция документа.
<Мин>	Значение типа «число», которое определяет минуту, на которую формируется позиция документа.
<Сек>	Значение типа «число», которое определяет секунду, на которую формируется позиция документа.
<ФлагКонцаСекунды>	Необязательный параметр. Число: 1 — позиция будет браться от следующей секунды. 0 — позиция от указанной секунды. Значение по умолчанию — 0.

Возвращаемое значение:

Строковое значение — сформированная позиция документа (см. «Позиция документа»).

Описание:

Функция СформироватьПозициюДокумента формирует и возвращает позицию документа согласно переданным параметрам.

Пример:

```
ПозицияВыбДокум=СформироватьПозициюДокумента(ВыбДокумент);
```

РазобратьПозициюДокумента

Раскладывает позицию документа на составляющие и возвращает строковое представление позиции документа

Синтаксис:

```
РазобратьПозициюДокумента(<Позиция>, <Дата>, <Час>, <Мин> , <Сек>, <Документ>)
```

Англоязычный синоним:

```
SplitDocPosition
```

Параметры:

<Позиция>	Строковое значение позиции документа.
<Дата>	Необязательный параметр. Имя переменной, куда система вернет значение даты позиции документа.
<Час>	Необязательный параметр. Имя переменной, куда система вернет числовое значение часа позиции документа.
<Мин>	Необязательный параметр. Имя переменной, куда система вернет числовое значение минуты позиции документа.
<Сек>	Необязательный параметр. Имя переменной, куда система вернет числовое значение секунды позиции документа.
<Документ>	Необязательный параметр. Имя переменной, куда система вернет значение документа данной позиции документа.

Возвращаемое значение:

Представление позиции документа в строковом виде, например: «12.07.98 12:57:14 ПрихНакл 24»

Описание:

Функция РазобратьПозициюДокумента раскладывает позицию документа на составляющие и возвращает строковое представление позиции документа.

Пример:

```
Позиция = РазобратьПозициюДокумента(ВыбДокумент.ПозицияДокумента());
```

Процедуры и функции форматирования

Пропись

Задать образец вывода числа прописью.

Синтаксис:

```
Пропись(<Образец>)
```

Англоязычный синоним:

```
Spelling
```

Параметры:

<Образец>	Необязательный параметр. В данном параметре можно передавать: -строковое выражение, задающее имя файла прописи (в поставляемом комплекте файлы прописей имеют расширение .SPL); - объект типа «СписокЗначений» в котором в специальном формате задан образец прописей. Если параметр <Образец> пустой или состоит из пробелов, то устанавливается образец прописей по умолчанию (файл 1CV7.spl).

Описание:

Процедура Пропись задает образец, в соответствии с которым будут выводиться прописные значения функцией Формат.

В параметре <Образец> может быть передана строка, задающая имя файла прописей. Файл прописей представляет собой текстовый файл специальной структуры, в котором находятся образцы прописей чисел, названия денежной единицы, дат, дней недели и т. п. Пример файла прописей можно посмотреть в поставляемом комплекте (файл lCV7.spl).

Ниже приводится фрагмент текста файла прописей, содержащий образец прописи для валюты мужского рода:

```
{"Speller",
{"Money",
{"Рубль", "Рубля", "Рублей", "Копейка", "Копейки", "Копеек", "М"}},
```

Ниже приводится фрагмент текста файла, содержащий образец прописи для валюты женского рода (добавляется параметр "F" в описание названия валюты):

```
{"Speller",
{"Money",
{"условная единица", "Условные единицы", "Условных единиц", "Сотая", "Сотые",
"Сотых", "F"}},
```

Кроме того, в параметре <Образец> может быть передан объект типа «СписокЗначений», содержащий в качестве своих значений объекты типа «Список-Значений» каждый из которых содержит набор слов используемых в прописи определенного типа. Типы прописей соответвуют секциям файла SPL:

```
Money
Numbers
Date
DateRange
WeekDay
```

Списки значений с наборами слов определенного типа прописей должны следовать либо в строго определенной последовательности, либо значение, являющееся списком значений, должно иметь в качестве своего строкового представления название типа прописи — в этом случае в переданном списке значений могут присутствовать не все списки значений соотвутствующие секциям файла SPL.

В списке значений конкретного типа прописи должны быть слова в последовательности, строго соответвующей последовательности слов в данной секции в файле SPL.

Действие данной процедуры распространяется только на текущий программный модуль.

Пример:

```
Процедура Печать(Тип)
   Таб = СоздатьОбъект("Таблица");
   // ...
   Пропись("1cue.spl");  // задаем имя файла прописи
   Таб.Опции(0, 0, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("Печать отчета", "");
```

```
    Пропись("");
    // возвращаем значение прописи по умолчанию
КонецПроцедуры
```

Формат

Форматировать переданный параметр.

Синтаксис:

`Формат(<Параметр>, <Форматная_строка>)`

Англоязычный синоним:

`Format`

Параметры:

`<Параметр>`	Выражение.
`<Форматная_строка>`	Строковое выражение.

Возвращаемое значение:

Строковое значение.

Описание:

Функция `Формат` форматирует переданный параметр, руководствуясь информацией, содержащейся в параметре `<Форматная_строка>`. Возвращает сформатированную строку.

Параметр может быть сформатирован как число, строка или дата. Если тип параметра не соответствует виду форматирования, указанному в форматной строке, то будет произведено соответствующее преобразование типа.

Форматная строка состоит из символа, определяющего тип данных, для которого будет производиться форматирование, и дополнительных параметров, уточняющих как должно быть сформатировано значение представленное первым параметром. Результатом работы функции является строковое значение, представляющее результаты форматирования.

Символы, определяющие тип форматируемых данных:

`Ч` — число (Англоязычный синоним: `N`)

`С` — строка (Англоязычный синоним: `S`)

`Д` — дата (Англоязычный синоним: `D`)

Между символом, определяющим тип форматируемого значения и дополнительными уточняющими параметрами может находиться любое число пробелов.

Для числовых значений форматная строка должна иметь вид: "`Чm.nDT`" или "`Чm`", где `т` и `п`— целые числа. Целое положительное число m определяет длину поля в котором будет размещаться строковое представление числа, а целое положительное число n определяет число знаков после десятичной точки. `D` — представляет собой любой символ (кроме пробела и цифры), которым в сфор-матированной строке целая часть будет отделяться от дробной, а `T` — символ (также кроме пробела и цифры), которым будут разделяться триады целой части числа. Символы "`D`" и "`T`" являются необязательными.

Если в форматной строке проставить символ "`0`" (ноль) перед длиной поля, то нулевые значения при выводе будут подавлятся (выводится пустая строка). Пример: "`Ч015.2`".

Если в форматной строке указано "`(0)`" перед всеми прочими спецификациями, то все позиции в поле вывода числа, соответствующие незадействованным старшим разрядам будут заполнены символами "`0`". Например:

```
Формат(123.15, "Ч(0)10.2") = 0000123.15
```

Если в форматной строке проставить символ "`-`" перед длиной поля, то нулевые значения при выводе будут отображаться прочерком. Пример: "`4-17.2 `".

В форматной строке для числа можно использовать сдвиг разрядов при выводе. Это позволяет, например, отображать числовое значение в тысячах и т. п. В этом случае форматная строка должна заканчиваться символами "`>X`" , где `X` — количество сдвигаемых разрядов. Например, форматная строка: "`Ч010>3`" — означает вывод числа в тысячах (сдвиг на три разряда).

Строковое представление числа всегда является правоустановленным.

Кроме того, возможен вывод денежных величин и просто целых чисел прописью. В этом случае форматная строка должна иметь вид "`ЧПДС`". Наличие буквы "`П`" (в английском варианте для аналогичной цели используется буква "`S`") определяет сам факт вывода числа прописью. Символ "`Д`" (в английском варианте — "`M`") обозначает вывод денежной величины и является необязательным. Символ "`С`" (в английском варианте — "`H`") также необязателен и означает вывод с копейками.

Для форматирования строк используется следующая форматная строка:

"`Cn`", где `n` — положительное целое число определяющее ширину поля, в котором будет размещаться строка. В случае, если ширина поля превосходит длину строки сформатированная строка будет дополнена справа пробелами, если же ширина поля меньше длины строки — строка будет усечена справа.

Для форматирования даты используется форматная строка вида "`Д<ПодстрокаФормата>`", где `<Подстрока Формата>` представляет собой строку, определяющую вид сформатированного представления даты:

`DDMMYY (ДДММГГ)`	дата в виде ДД.ММ.ГГ
`DDMMYYYY`	дата в виде ДД.ММ.ГГГГ

(ДДММГГГГ)	
DDMMMMYYYY	дата в виде ДД месяц прописью ГГГГ
(ДДММММГГГГ)	
(0)DDMMMMYYYY	этот формат представляет собой вариант предыдущего формата с тем отличием, что чис-
((0)ДДММММГГГГ)	ло месяца всегда выводится двумя цифрами, т. е. для чисел меньших 10 спереди будет добавлен 0. Например: Формат('01.01.1999', ' 'Д(0)ДДММММГГГГ' ') = 01 Января 1999 г.
MMMMYYYY	дата в виде месяц прописью ГГГГ
(ММММГГГГ)	
MMMMYY (ММММГГ)	дата в виде месяц прописью ГГ
MMMM (ММММ)	дата в виде месяц прописью
QQQQYYYY	дата в виде N квартала ГГГГ
(ККККГГГГ)	
QQQQYY (ККККГГ)	дата в виде N квартала ГГ
QQQQ	дата в виде N квартала
YYYYMMDD	дата в виде ГГГГММДД, то есть дата '10.11.1998' будет представлена как строка
(ГГГГММДД)	"19981110"
WWWW (НННН)	выводит наименование дня недели, соответствующее указанной дате. Наименования дней недели берутся из файла прописи.

Пример:

```
ДолгКлиента = Строка(Формат(-Долг(), "Ч12.2")) + " " + Валюта.Сокр_назв;
```

Шаблон

Сформировать строку по шаблону.

Синтаксис:

```
Шаблон(<Строка_шаблон>)
```

Англоязычный синоним:

```
Template
```

Параметры:

<Строка_шаблон>	Строковое выражение, содержащее шаблон формирования результирующей строки.

Возвращаемое значение:

Строковое значение.

Описание:

Функция Шаблон возвращает строковое значение, сформированное в соответствии с заданным параметром <Строка_шаблон>, заменив в ней все встроенные выражения, заключенные в квадратные скобки, на их строковые значения.

Строковое значение параметра <Строка_шаблон> обрабатывается следующим образом: символы "[" , "]" (квадратные скобки) являются специальными символами, которые выделяют поля, содержащие встроенные выражения. Например:

```
Имя клиента [Имя], телефон: [Телефон]
```

После встроенного выражения может стоять символ "#", после которого следует Форматная_строка описанная в функции Формат. В этом случае в данное поле будет проставлен результат выражения, обработанный функцией Формат.

Пример:

```
Стр = Шаблон("Это название услуги: [Услуга.Наименование]");
//   выдает: "Это название услуги: Вывоз мусора"
```

ФиксШаблон

Сформировать строку по фиксированному шаблону.

Синтаксис:

```
ФиксШаблон(<Строка_шаблон>)
```

Англоязычный синоним:

```
FixTemplate
```

Параметры:

<Строка_шаблон>	Строковое выражение, содержащее шаблон формирования результирующей строки.

Возвращаемое значение:

Строковое значение.

Описание:

Функция ФиксШаблон возвращает строковое значение, сформированное в соответствии с заданным параметром <Строка_шаблон>, заменив в ней все встроенные выражения, заключенные в квадратные скобки, на их значения. В отличие от функции Шаблон, поля замещаются значениями выражений с сохранением своей длины в символах, то есть обрезаются, если поле короче результата вычисления выражения и дополняются пробелами если длиннее. Если результат числовой, то в границах поля строка прижимается к правой границе.

Строковое значение параметра <Строка_шаблон> обрабатывается следующим образом: символы "[" , "]" (квадратные скобки) являются специальными символами, которые выделяют поля, содержащие встроенные выражения. Выделенные поля фиксируют свою длину с учетом ограничивающих скобок. Например:

```
Имя Клиента [Имя], телефон: [Телефон]
```

После встроенного выражения может стоять символ "#", после которого следует Форматная_строка описанная в функции Формат. В этом случае в данное поле будет проставлен результат выражения, обработанный функцией Формат.

Пример:

```
// ниже - пример с дополнением пробелами
Стр = Шаблон("Это название услуги [Услуга.Наименование       ]");
//   выдает: "Это название услуги Вывоз мусора                "
// далее - с сокращением строки
Стр = Шаблон("Это название услуги [Услуга.Наименование]");
//   выдает: "Это название услуги Выполенние формирован"
```

Функции для вызова диалога ввода данных

ВвестиЗначение

Вызов диалога для ввода значения заданного типа.

Синтаксис:

```
ВвестиЗначение(<Значение>, <Подсказка>, <Тип>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
InputValue
```

Параметры:

<Значение>	Имя переменной, ранее объявленной в программном модуле. В эту переменную будет помещено введенное значение.
<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.
<Тип>	Строковое выражение — название типа данных, которое требуется ввести. Например: "Строка", "Число", "Справочник.Товары", "Документ . РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Длина вводимого значения (для типов данных «Строка», «Число»).
<Точность>	Необязательный параметр. Количество знаков после десятичной точки (для типа данных «Число»).

Возвращаемое значение:

Числовое значение: 1 — если в диалоге нажата кнопка «ОК»; 0 — если нажата кнопка «Отмена»;

Описание:

Функция ВвестиЗначение выполняет вызов диалога для ввода значения заданного типа.

Пример:

```
Если ВвестиЗначение(Ном, "Введите номер приказа", "Число", 5, 0) = 1 Тогда
   ВыбДок = СоздатьОбъект("Документ.Приказ");
   ВыбДок.НайтиПоНомеру(Строка(Ном), Дата(0));
   Если ВыбДок.Выбран() = 1 Тогда
      // . . .
   ИначеЕсли;
      // . . .
   КонецЕсли;
КонецЕсли;
```

ВвестиЧисло

Вызов диалога для ввода числа.

Синтаксис:

```
ВвестиЧисло(<Число>, <Подсказка>, <Длина>, <Точность> , <Таймаут>)
```

Англоязычный синоним:

```
InputNumeric
```

Параметры:

<Число>	Имя переменной, ранее объявленной в программном модуле. В эту переменную будет помещено введенное значение числа.
<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.

<Длина>	Длина вводимого числа.
<Точность>	Количество знаков после десятичной точки.
<Таймаут>	Необязательный параметр. Числовое выражение интервала времени ожидания ответа в секундах, в течение которого система будет ожидать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Возвращаемое значение:

Числовое значение: 1 — если в диалоге нажата кнопка «ОК»; 0 — если нажата кнопка «Отмена»; -1 (минус единица) — если закончилось время ожидания ответа.

Описание:

Функция ВвестиЧисло выполняет вызов диалога для ввода числа.

Пример:

```
Если ВвестиЧисло(Ном, "Введите номер приказа", 5, 0, 10) = 1 Тогда
   ВыбДок = СоздатьОбъект("Документ.Приказ");
   ВыбДок.НайтиПоНомеру(Строка(Ном), Дата(0));
   Если ВыбДок.Выбран() = 1 Тогда
      // . . .
   ИначеЕсли;
      // . . .
   КонецЕсли;
КонецЕсли;
```

ВвестиСтроку

Вызов диалога для ввода строки.

Синтаксис:

```
ВвестиСтроку(<Строка>, <Подсказка>, <ДлинаСтроки>, <Признак>, <Таймаут>)
```

Англоязычный синоним:

```
InputString
```

Параметры:

<Строка>	Имя переменной, ранее объявленной в программном модуле. В эту переменную будет помещено введенное строковое значение.
<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.
<ДлинаСтроки>	Числовое выражение — длина строки.
<Признак>	Необязательный параметр. Числовое выражение, если 0 — ввод простой строки без разделителей строк, если 1 — ввод многострочного текста с разделителями строк. Значение по умолчанию — 0.
<Таймаут>	Необязательный параметр. Числовое выражение интервала времени ожидания ответа в секундах, в течение которого система будет ожидать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Возвращаемое значение:

Числовое значение: 1 — если в диалоге нажата кнопка «ОК», 0 — если нажата кнопка «Отмена»; -1 (минус единица) — если закончилось время ожидания ответа.

Описание:

Функция ВвестиСтроку выполняет вызов диалога для ввода строки.

Пример:

```
Процедура ВводКодовПродукции()
   Если Продукция.Вид = Перечисление.ВидПродукции.Наша Тогда
      НК = Код;
      Если ВвестиСтроку(НК, "Введите код продукции", 40, 1, 9) = 1 Тогда
         Код = НК;
      КонецЕсли;
   Иначе
      Код = "";
      Возврат;
   КонецЕсли;
КонецПроцедуры
```

ВвестиДату

Вызов диалога для ввода даты.

Синтаксис:

```
ВвестиДату(<Дата>, <Подсказка>, <Таймаут>)
```

Англоязычный синоним:

```
InputDate
```

Параметры:

<Дата>	Имя переменной, ранее объявленной в программном модуле. В эту переменную будет помещено введенное значение даты.
<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.
<Таймаут>	Необязательный параметр. Числовое выражение интервала времени ожидания ответа в секундах, в течение которого система будет ожидать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Возвращаемое значение:

Числовое значение: 1 — если в диалоге нажата кнопка «ОК»; 0 — если нажата кнопка «Отмена»; -1 (минус единица) — если закончилось время ожидания ответа.

Описание:

Функция ВвестиДату выполняет вызов диалога для ввода даты.

Пример:

```
Процедура ВводДатыССуды()
   Если ВидРасчета = ВидРасчета.БеспроцентнаяССуда Тогда
      ДатаССуды = Дата(0);
      Возврат;
   КонецЕсли;
   Если ДатаССуды = Дата(0) Тогда
      ДатаССуды = ДатаДок + 30;
   КонецЕсли;
   Д = ДатаССуды;
   Если ВвестиДату(Д, "Введите дату ссуды") = 1 Тогда
      ДатаССуды = Д;
   Иначе
      ВидРасчета = ВидРасчета.БеспроцентнаяССуда;
   КонецЕсли;
КонецПроцедуры
```

ВвестиПериод

Ввод периода в виде диалога.

Синтаксис:

```
ВвестиПериод(<НачалоПериода>, <КонецПериода>, <Подсказка>)
```

Англоязычный синоним:

```
InputPeriod
```

Параметры:

<НачалоПериода>	Идентификатор переменной, в которую функция возвращает дату начала периода.
<КонецПериода>	Идентификатор переменной, в которую функция возвращает дату конца периода.
<Подсказка>	Необязательный параметр. Строковое выражение, содержащее строку, которая будет выводиться в заголовке окна диалога. Если не указано — стандартная подсказка.

Возвращаемое значение:

Числовое значение: 1 — выбор осуществлен , 0 — выбор не осуществлен (пользователем нажата кнопка «Отмена», клавиша <Esc> или закрыто окно диалога).

Описание:

Функция ВвестиПериод выдает на экран диалог специального вида, при помощи которого пользователь может выбрать период. Выбранный пользователем период записывается в переменные, передаваемые как параметры при вызове метода.

Пример:

```
Дата1 = '01.01.97';
Дата2 = '31.03.97';
Если ВвестиПериод(Дата1, Дата2, "Введите период отчета") <> 1 Тогда
   Возврат 0;
КонецЕсли;
```

ВвестиПеречисление

Вызов диалога для ввода перечисления.

Синтаксис:

ВвестиПеречисление(<Значение>, <Подсказка>, <Таймаут>)

Англоязычный синоним:

InputEnum

Параметры:

<Значение>	Имя переменной, ранее объявленной в программном модуле. Переменная должна содержать значение типа «Перечисление» или строковое значение. Если тип передаваемого в качестве параметра значения — перечисление, то вызывается диалог со списком заданного вида перечисления. Если тип передаваемого в качестве параметра значения — строка, то в ней должен быть идентификатор требуемого вида перечисления, как он задан в конфигураторе. В эту же переменную будет помещено выбранное в диалоге значение перечисления.
<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.
<Таймаут>	Необязательный параметр. Числовое выражение интервала времени ожидания ответа в секундах, в течение которого система будет ожидать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Возвращаемое значение:

Числовое значение: 1 — если в диалоге нажата кнопка «ОК», 0 — если нажата кнопка «Отмена»; -1 (минус единица) — если закончилось время ожидания ответа.

Описание:

Функция ВвестиПеречисление выполняет вызов диалога для ввода перечисления.

Пример:

```
*
Процедура УстПризнак1()
   ВыбПризн = Перечисление.ТипСотрудника.Штатный;
   Если ВвестиПеречисление(ВыбПризн, "Выберите тип") > 0 Тогда
      Тип = ВыбПризн;
   Иначе
      Тип = Перечисление.ТипСотрудника.Штатный;
   КонецЕсли;
КонецПроцедуры

*
Процедура УстПризнак2()
   ВыбПризн = "ТипСотрудника";
   Если ВвестиПеречисление(ВыбПризн, "Выберите тип") > 0 Тогда
      Тип = ВыбПризн;
   Иначе
      Тип = Перечисление.ТипСотрудника.Штатный;
   КонецЕсли;
КонецПроцедуры
```

Процедуры и функции общего назначения

Вопрос

Вывести окно вопроса.

Синтаксис:

Вопрос(<Текст_вопроса>, <Режим>, <Таймаут>)

Англоязычный синоним:

DoQueryBox

Параметры:

<Текст_вопроса>	Строковое выражение текста вопроса.
<Режим>	Числовое или строковое выражение, определяющее режим вывода окна вопроса.
<Таймаут>	Необязательный параметр. Числовое выражение интервала времени ожидания ответа в секундах, в течение которого система будет ожидать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Возвращаемое значение:

Если параметр <Режим> задан числовым значением, то функция возвращает числовое значение:

-1	Закончилось время ожидания ответа.
1	Выбрана кнопка «ОК».

2 Выбрана кнопка «Отмена».
3 Выбрана кнопка «Стоп».
4 Выбрана кнопка «Повтор».
5 Выбрана кнопка «Пропустить».
6 Выбрана кнопка «Да».
7 Выбрана кнопка «Нет».

Если параметр <Режим> задан строковым значением, то функция возвращает строковое значение. Язык для возврата такой же, какой использован в пара-метре <Режим>.

Русск. яз.	Англ.яз.	Описание
Таймаут	Timeout	Закончилось время ожидания ответа.
ОК	OK	Выбрана кнопка «ОК».
Отмена	Cancel	Выбрана кнопка «Отмена».
Стоп	Abort	Выбрана кнопка «Стоп».
Повтор	Retry	Выбрана кнопка «Повтор».
Пропустить	Ignore	Выбрана кнопка «Пропустить».
Да	Yes	Выбрана кнопка «Да».
Нет	No	Выбрана кнопка «Нет».

Описание:

Функция Вопрос выводит на экран окно вопроса. Текст определяется параметром <Текст_вопроса>. Параметр <Режим> определяет варианты возможных ответов.

Если параметр <Режим> задан числовым значением, то возможные варианты передаваемого параметра:

0 Кнопка «ОК».
1 Кнопки «ОК» и «Отмена».
2 Кнопки «Стоп», «Повтор», «Пропустить».
3 Кнопки «Да», «Нет», «Отмена».
4 Кнопки «Да», «Нет».
5 Кнопки «Повтор», «Отмена».

Любое другое числовое значение параметра <Режим> эквивалентно значению 0 (кнопка «ОК»), равно как и отсутствие указанного параметра.

Если параметр <Режим> задан строковым значением, то возможные варианты передаваемого параметра:

Русскоязычное написание	Англоязычное написание
ОК	OK
ОК+Отмена	OK+Cancel
Стоп+Повтор+Пропустить	Abort+Retry+Ignore
Да+Нет+Отмена	Yes+No+Cancel
Да+Нет	Yes+No
Повтор+Отмена	Retry+Cancel

Пример:

```
Рез = Вопрос("Вы получили данное сообщение?", 4);
Если Вопрос("Все нормально?", "Да+Нет") = "Да" Тогда
   //  . . .
КонецЕсли;
```

Предупреждение

Вывести окно предупреждения.

Синтаксис:

Предупреждение(<Текст_сообщения>, <Таймаут>)

Англоязычный синоним:

DoMessageBox

Параметры:

<Текст_сообщения> Строковое выражение.

<Таймаут> Необязательный параметр. Числовое выражение интервала времени ожидания в секундах, в течение которого система будет ждать ответа пользователя. Если данный параметр опущен или равен 0, то время ожидания бесконечно. Значение по умолчанию — 0.

Описание:

Процедура Предупреждение выводит на экран окно предупреждения (MessageBox). Текст определяется параметром <Текст_сообщения>. По поведению данная процедура эквивалентна функции:

Вопрос(<Текст_сообщения>, 0, <Таймаут>), но не возвращает значения.

Если закончилось время ожидания ответа, то окно предупреждения автоматически удаляется с экрана.

Пример:

```
// Выводим сообщение 5 секунд
Предупреждение("Доброе утро!", 5);
```

Сообщить

Вывести строку в окно сообщений.

Синтаксис:

```
Сообщить(<Текст_сообщения>, <ИмиджМаркера>)
```

Англоязычный синоним:

```
Message
```

Параметры:

<Текст_сообщения>	Строковое выражение.
<ИмиджМаркера>	Необязательный параметр. Строковое выражение, которое задает тип пиктограммы выводимой перед сообщением. Возможные значения: "I", "!", "!!", "!!!", "." — обычное сообщение, " " — без маркера.

Описание:

Процедура Сообщить выводит <Текст_сообщения> в окно сообщений. Перед сообщениями можно отображать специальные пиктограммы, которыми можно помечать сообщения различной важности.

Пример:

```
Сообщить("Доброе утро!", "I");
```

ОчиститьОкноСообщений

Очистить окно сообщений.

Синтаксис:

```
ОчиститьОкноСообщений()
```

Англоязычный синоним:

```
ClearMessageWindow
```

Описание:

Процедура ОчиститьОкноСообщений очищает окно сообщений.

Пример:

```
ОчиститьОкноСообщений();
```

Состояние

Вывести сообщение в строку состояния.

Синтаксис:

```
Состояние(Текст_сообщения)
```

Англоязычный синоним:

```
Status
```

Параметры:

<Текст_сообщения>	Строковое выражение.

Описание:

Процедура Состояние выводит строку текста в строку состояния (статус-бар). Текст определяется параметром <Текст_сообщения>.

Пример:

```
Состояние("Доброе утро!")
```

? (вычислить выражение по условию)

Вычислить выражение по условию.

Синтаксис:

```
?(<Логич_выраж>, <Выраж1>, <Выраж2>)
```

Англоязычный синоним:

```
?
```

Параметры:

<Логич_выраж>	Логическое выражение.
<Выраж1>	Выражение.

<Выраж2> Выражение.

Возвращаемое значение:
Результирующее значение.
Описание:
Функция ? вычисляет значение <Логич_выраж> и если его значение — истина, то возвращает вычисленное значение <Выраж1>. Если значение <Логич_выраж> — ложь, то возвращает вычисленное значение <Выраж2>.
Пример:

```
Спр.ВыбратьЭлементы();
Пока (Спр.ПолучитьЭлемент() > 0) Цикл
   Состояние(Спр.Наименование);
   Сообщить(Спр.Наименование);
   Sec = ?(Спр.ЭтоГруппа = 1, "Folder", "DL");
   Таб.ВывестиСекцию(Sec + "-V1");
   Таб.ПрисоединитьСекцию(Sec + "-V3");
КонецЦикла;
```

Сигнал

Вывести звуковой сигнал.
Синтаксис:
Сигнал()
Англоязычный синоним:
Beep
Описание:
Процедура Сигнал издает короткий звуковой сигнал.
Пример:

```
Сигнал();
```

Разм

Определить размерность массива.
Синтаксис:
Разм(<Имя_массива>)
Англоязычный синоним:
Dim
Параметры:
<Имя_массива> Идентификатор массива.
Возвращаемое значение:
Числовое значение размерности массива.
Описание:
Функция Разм возвращает числовое значение размерности массива переданного в качестве параметра.
Пример:

```
Перем Массив_для_хранения[10];
Размер = Разм(Массив_для_хранения);
```

Функции среды исполнения

ЗаголовокСистемы

Получить/установить заголовок окна программы.
Синтаксис:
ЗаголовокСистемы(<Заголовок>)
Англоязычный синоним:
SystemCaption
Параметры:
<Заголовок> Строковое выражение.
Возвращаемое значение:
Строковое значение — заголовок системы до исполнения метода.
Описание:
Метод ЗаголовокСистемы позволяет получить/установить заголовок окна программы.
Пример:

```
ЗаголовокСистемы("Оптово-розничная конфигурация")
```

ИмяКомпьютера

Получить сетевое имя компьютера.
Синтаксис:
ИмяКомпьютера()
Англоязычный синоним:
ComputerName
Возвращаемое значение:
Строковое значение — сетевое имя компьютера, работающего в данный момент с программой.
Описание:
Функция ИмяКомпьютера возвращает сетевое имя компьютера, работающего в данный момент с программой.
Пример:
```
Компьютер = ИмяКомпьютера();
```

ИмяПользователя

Получить имя пользователя.
Синтаксис:
ИмяПользователя()
Англоязычный синоним:
UserName
Возвращаемое значение:
Строковое значение — имя пользователя, работающего в данный момент с программой.
Описание:
Функция ИмяПользователя возвращает имя пользователя (указанное в конфигураторе в списке пользователей), работающего в данный момент с программой.
Пример:
```
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.НайтиПоКоду(ИмяПользователя());
Менеджер = Спр.ТекущийЭлемент();
Если Менеджер.Выбран() = 0 Тогда
   Предупреждение("Не выбран менеджер!");
КонецЕсли;
```

ПолноеИмяПользователя

Получить полное имя пользователя.
Синтаксис:
ПолноеИмяПользователя()
Англоязычный синоним:
UserFullName
Возвращаемое значение:
Строковое значение, содержащее полное имени пользователя.
Описание:
Функция ПолноеИмяПользователя возвращает полное имя пользователя, указанное в конфигураторе при авторизации доступа.
Пример:
```
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.НайтиПоНаименованию(ПолноеИмяПользователя());
Менеджер = Спр.ТекущийЭлемент();
Если Менеджер.Выбран() = 0 Тогда
   Предупреждение("Не выбран менеджер!");
КонецЕсли;
```

НазваниеНабораПрав

Получить название набора прав пользователя.
Синтаксис:
НазваниеНабораПрав(<ВыдИспНабПрав>)
Англоязычный синоним:
RightName
Параметры:

<ВыдИспНабПрав>	Выдавать используемый набор прав или нет. Необязательный параметр. Число: 0 — выдает установленный пользователю набор прав; 1 — выдает пустую строку, если пользо-

вателю набор прав назначен, но установлен режим отключения контроля набора прав. Значение по умолчанию — 0.

Возвращаемое значение:
Строковое значение, содержащее название набора прав пользователя.
Описание:
Функция НазваниеНабораПрав возвращает название набора прав, заданное пользователю в конфигураторе.
Пример:
```
МоиПрава = НазваниеНабораПрав();
```

ПравоДоступа

Проверяет для текущего пользователя наличие права доступа для заданного объекта.
Синтаксис:
```
ПравоДоступа(<НазваниеПрава>, <Объект>)
```
Англоязычный синоним:
```
AccessRight
```
Параметры:

<НазваниеПрава>	Строка с названием права доступа, как оно выводится в конфигурации (без пробелов).
<Объект>	Строка с наименованием типа и вида объекта (записывается через точку). Для глобальных прав этот параметр не указывается. Для объектов, у которых нет вида, записывается только тип объекта, например, «Операция».

Возвращаемое значение:
Число: 1 — если право доступа есть, иначе 0.
Описание:
Функция ПравоДоступа проверяет для текущего пользователя наличие права доступа для заданного объекта.
Пример:
```
Доступ = ПравоДоступа("Чтение", "Документ.Счет");
Доступ1 = ПравоДоступа("МонопольныйРежим");
```

НазваниеИнтерфейса

Получить название интерфейса пользователя.
Синтаксис:
```
НазваниеИнтерфейса()
```
Англоязычный синоним:
```
UserInterfaceName
```
Возвращаемое значение:
Строковое значение, содержащее название интерфейса пользователя.
Описание:
Функция НазваниеИнтерфейса возвращает название интерфейса, заданное пользователю в конфигураторе.
Пример:
```
МойИнтерфейс = НазваниеИнтерфейса();
```

КаталогПользователя

Получить каталог пользователя.
Синтаксис:
```
КаталогПользователя()
```
Англоязычный синоним:
```
UserDir
```
Возвращаемое значение:
Строковое значение, содержащее имя рабочего каталога пользователя.
Описание:
Функция КаталогПользователя возвращает имя рабочего каталога пользователя, заданный пользователю в конфигураторе.
Пример:
```
МояДиректория = КаталогПользователя();
```

КаталогИБ

Получить каталог базы данных.
Синтаксис:
```
КаталогИБ()
```
Англоязычный синоним:

IBDir

Возвращаемое значение:

Строковое значение, содержащее имя каталога базы данных.

Описание:

Функция КаталогИБ возвращает имя каталога базы данных.

Пример:

```
МояБД = КаталогИБ();
```

КаталогПрограммы

Получить каталог программы.

Синтаксис:

КаталогПрограммы()

Англоязычный синоним:

BinDir

Возвращаемое значение:

Строковое значение, содержащее имя каталога программы.

Описание:

Функция КаталогПрограммы возвращает имя каталога, где размещены исполняемые файлы системы 1С:Предприятие.

Пример:

```
МояДирПрог = КаталогПрограммы();
```

КаталогВременныхФайлов

Получить каталог временных файлов.

Синтаксис:

КаталогВременныхФайлов()

Англоязычный синоним:

TempFilesDir

Возвращаемое значение:

Строковое значение, содержащее имя каталога временных файлов.

Описание:

Функция КаталогВременныхФайлов возвращает имя каталога временных файлов, как он установлен в системе 1С:Предприятие. Это может быть каталог, установленный в командной строке запуска системы или, если не указан, то каталог временных файлов установленный в операционной системе.

Пример:

```
МояВремДир = КаталогВременныхФайлов();
```

МонопольныйРежим

Определение режима работы программы.

Синтаксис:

МонопольныйРежим()

Англоязычный синоним:

ExclusiveMode

Возвращаемое значение:

Число 1 — если программа запущена в монопольном режиме;
Число 0 — если программа запущена в сетевом режиме.

Описание:

Функция МонопольныйРежим возвращает число 1 — если программа запущена в монопольном режиме и 0 если нет.

Пример:

```
Процедура Загрузка()
   Если МонопольныйРежим() > 0 Тогда
      ЗагрузитьНаряды();
   Иначе
      Предупреждение("Для запуска этой операции требуется" +
                     "монопольный режим доступа !!!");
   КонецЕсли;
КонецПроцедуры
```

ОсновнойЯзык

Определение основного языка конфигурации.

Синтаксис:
`ОсновнойЯзык()`
Англоязычный синоним:
`GeneralLanguage`
Возвращаемое значение:
Число 1 — если основной язык конфигурации — русский;
Число 0 — если основной язык конфигурации — английский.
Описание:
При помощи функции `ОсновнойЯзык` можно прочитать текущее значение основного языка конфигурации.

Замечание: Значение основного языка конфигурации задается в конфигураторе (*Метаданные-Глобалъный_модуль-Свойство-Задача-Основной язык*).

Значение основного языка конфигурации используется системой в нескольких случаях, а именно, когда система передает в качестве параметра в предопределенную процедуру название объекта конфигурации. В этом случае название агрегатного типа данных передается либо в русском либо в английском написании — в зависимости от текущей установки основного языка конфигурации.

Пример:
*

В данном примере при записи периодического реквизита справочника (через диалоговое окно «История») система передает в предопределенную процедуру строковое значение названия объекта. Нижеследующая процедура определяет, что введено значение курса валюты и сообщает об этом.

```
Процедура ПриЗаписиИстории(ИмяОбъекта, Объект, Значение, ДатаИстории)
   Если ОсновнойЯзык() = 1 Тогда
      СПР = "Справочник";
   Иначе
      СПР = "Reference";
   КонецЕсли;
   Если ИмяОбъекта = СПР + ".Валюты.Текущ_курс" Тогда
      Сообщить("Добавлено новое значение курса");
      Сообщить("Для валюты " + Объект.Наименование);
      Сообщить("На дату " + ДатаИстории);
      Сообщить("Установлено значение " + Значение);
   КонецЕсли;
КонецПроцедуры
```

См. также: `ПриУдаленииИстории, ПриЗаписиИстории`

Процедуры работы с транзакциями

Понятие транзакций соответствует общепринятому пониманию транзакций баз данных. В системе 1С:Предприятие транзакции активно используются самой системой при записи различной информации. Транзакция гарантирует неизменность информации в базе данных для других пользователей до ее завершения и целостное сохранение информации.

Возможность использования транзакций в языке должна применяться только в тех программных модулях, которые выполняют запись изменений в информационной базе (в справочниках, константах, документах). Их не следует использовать в алгоритмах формирования движений документов. Транзакции также не стоит использовать при одиночных записях. Типичный пример когда следует использовать транзакции — это процедура, которая будет во всех элементах справочника менять некоторый параметр.

Для начала транзакции используется процедура `НачатьТранзакцию`. Для фиксации сделанных в процессе выполнения изменений применяется процедура `ЗафиксироватьТранзакцию`. Для отмены изменений, сделанных в процессе выполнения транзакции — процедура `ОтменитьТранзакцию`. Таким образом, все действия с базой данных, выполняемые внутри скобок: `НачатьТранзакцию — ЗафиксироватьТранзацию` собственно и являются транзакцией.

Действия, заключенные в транзакцию, выполняются быстрее. Особенно существенно разница проявляется в многопользовательском режиме и особенно существенна разница для операций, связанных с изменением содержимого базы данных, хотя и для операций только чтения разница может быть в разы.

Замечание. В Модуле документа (при проведении документов) в предопределенных процедурах `ОбработкаПроведения` и `ОбработкаУдаленияПроведения` система сама по умолчанию выполняет все действия через транзакцию, поэтому никаких специальных действий в этих предопределенных процедурах предпринимать не нужно.

Транзакция, выполняемая одним пользователем может мешать выполнению транзакций другими пользователями. Таким образом, важно соизмерять получающееся ускорение от применения транзакций с возможными побочными эффекта-

ми. Другими словами, возможно, что ускорение таково, что операция начинает выполняться столь быстро, что можно пренебречь влиянием на других пользователей (например, ожидание секунду – другую), а может это будет приводить к конфликтам — все зависит от конкретного алгоритма конфигурации.

При обработке транзакции (в том числе в Модуле документа) не следует использовать элементы интерактивного управления (например, операторы `Предупреждение`, `Вопрос`, `ВвестиЧисло` и т. п.), т. к. в этом случае при открытой транзакции система ожидает отклика пользователя, а это может препятствовать нормальной работе других пользователей (в результате документы у всех остальных пользователей в этот момент могут не проводиться). Если в конфигурации необходимо при проведении документа выдавать пользователю некоторые сообщения, то следует использовать операторы `Сообщить` или `Состояние`.

НачатьТранзакцию

Открыть обработку транзакции.
Синтаксис:
`НачатьТранзакцию()`
Англоязычный синоним:
`BeginTransaction`
Описание:
Процедура `НачатьТранзакцию` открывает транзакцию для обработки информации.
Пример:

```
Процедура УдалитьПустые()
   Спр = СоздатьОбъект("Справочник.Подразделения");
   Спр.ВыбратьЭлементы();
   НачатьТранзакцию();
   Пока Спр.ПолучитьЭлемент() = 1 Тогда
      Если Спр.Количество = 0 Тогда
         Спр.Удалить();
      КонецЕсли;
   КонецПока;
   ЗафиксироватьТранзакцию();
КонецПроцедуры
```

ЗафиксироватьТранзакцию

Завершить успешную транзакцию.
Синтаксис:
`ЗафиксироватьТранзакцию()`
Англоязычный синоним:
`CoinmitTransation`
Описание:
Процедура `ЗафиксироватьТранзакцию` завершает успешную транзакцию.
Пример:

```
Процедура УдалитьПустые()
   Спр = СоздатьОбъект("Справочник.Подразделения");
   Спр.ВыбратьЭлементы();
   НачатьТранзакцию();
   Пока Спр.ПолучитьЭлемент() = 1 Тогда
      Если Спр.Количество = 0 Тогда
         Спр.Удалить();
      КонецЕсли;
   КонецПока;
   ЗафиксироватьТранзакцию();
КонецПроцедуры
```

ОтменитьТранзакцию

Завершить транзакцию без записи изменений.
Синтаксис:
`ОтменитьТранзакцию()`
Англоязычный синоним:
`RollBackTransaction`
Описание:
Процедура `ОтменитьТранзакцию` завершает транзакцию без записи изменений.
Пример:

*

В данном примере приведена процедура заполнения справочника валют значениями курсов из внешнего текстового файла. В процедуре используется обработка транзакции, причем, если в процессе загрузки обнаружилось наличие некорректных данных во внешнем файле, то процедура завершается с отменой транзакции.

```
Процедура ЗагрузкаКурсовВалют()
   Влт = СоздатьОбъект("Справочник.Валюты");
   Текст = СоздатьОбъект("Текст");
   Текст.Открыть(ИмяФайла);
   Если Текст.КоличествоСтрок() = 0 Тогда
      Сообщить("Текст пустой!");
      Возврат;
   КонецЕсли;
   Успешно = 1;
   НачатьТранзакцию();
   Для Ном = 1 по Текст.КоличествоСтрок() Цикл
      Стр = Текст.ПолучитьСтроку(Ном);
      Если СтрДлина(Стр) = 0 Тогда
         Продолжить;
      КонецЕсли;
      Поле=0;
      Пока СтрДлина(Стр) > 0 Цикл
         Поле = Поле + 1;
         Инд = Найти(Стр, "-");
         Если Инд > 0 Тогда
            Если Инд > 3 Тогда
               Стр1 = Сред(Стр, 2, Инд-3);
               // не берем кавычки спереди и сзади, и символ тильды.
            Иначе
               // если значение поля пропущено
               Стр1 = "";
            КонецЕсли;
            Стр=Сред(Стр, Инд + 1);
            // не берем символ тильды
         Иначе
            Стр = Сред(Стр, 2);
            Инд = Найти(Стр, """");
            Стр1 = Сред(Стр, 1, Инд-1);
           Стр = "";
         КонецЕсли;
         Если Поле = 1 Тогда
            Код = Стр1;
         ИначеЕсли Поле = 2 Тогда
            Курс = Стр1;
         ИначеЕсли Поле = 3 Тогда
            ДатаКурса = Стр1;
         КонецЕсли;
      КонецЦикла;
      Влт.ИспользоватьДату(Дата(ДатаКурса));
      Влт.НайтиПоКоду(Код);
      Если Влт.Выбран() > 0 Тогда
         Сообщить(Формат(Ном, "45") + " - загрузка Курса="
              + Формат(Курс, "45") + " от " + ДатаКурса);
         Влт.Текущ_курс = Число(Курс);
         Влт.Записать();
         Сообщить("    - Загружен!");
      Иначе
         Успешно = 0;
         Сообщить("В базе данных нет валюты с кодом " + Код");
         Сообщить("Импорт данных отменён!");
         Прервать;
      КонецЕсли;
      Состояние("Загружено "+Ном+" строк.");
```

```
   КонецЦикла;
   Если Успешно = 1 Тогда
      ЗафиксироватьТранзакцию();
   Иначе
      ОтменитьТранзакцию();
   КонецЕсли;
КонецПроцедуры
```

Специальные процедуры и функции

СоздатьОбъект

Создает объект агрегатного типа данных и возвращает ссылку на него.

Синтаксис:

```
СоздатьОбъект(<ИмяАгрегатногоТипа>)
```

Англоязычный синоним:

```
CreateObject
```

Параметры:

<ИмяАгрегатногоТипа>	Строковое выражение, значение которого содержит имя агрегатного типа данных, объявленного в конфигураторе.

Возвращаемое значение:

Ссылка на созданный объект агрегатного типа данных.

Описание:

Функция `СоздатьОбъект` создает объект агрегатного типа данных и возвращает ссылку на него. Данная функция обычно используется одновременно с неявным определением переменной и присвоением ей ссылки на объект агрегатного типа данных.

Замечание. Объекты, такие как документ и справочник, созданные при помощи функции `СоздатьОбъект`, изначально не определены, т. е. не содержат никакого значения. Чтобы начать с ними работать, их предварительно надо позиционировать (установить на конкретный документ или элемент справочника). Для документов позиционирование выполняется при помощи методов `НайтиДокумент`, `НайтиПоНомеру`, `ПолучитьДокумент`. Для справочников позиционирование выполняется при помощи методов `НайтиЭлемент`, `НайтиПоКоду`, `ПолучитьЭлемент`.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.НайтиПоКоду("029");
```

СтатусВозврата

Установить/определить статуса возврата предопределенной процедуры.

Синтаксис:

```
СтатусВозврата([<Статус>])
```

Англоязычный синоним:

```
ReturnStatus
```

Параметры:

<Статус>	Необязательный параметр. Числовое выражение. Если задано значение 1, то устанавливается статус возврата — *Выполнить Действие*. Если задано значение 0, то устанавливается статус возврата — *Отменить Действие*. Если параметр опущен, то статус возврата предопределенной процедуры не меняется.

Возвращаемое значение:

Число 1 — если статус возврата — *Выполнить Действие*;
Число 0 — если статус возврата — *Отменить Действие*.

Описание:

Применять функцию `СтатусВозврата` имеет смысл только в теле предопределенных процедур. При помощи функции `СтатусВозврата` можно установить или прочитать текущее значение статуса возврата конкретной предопределенной процедуры, где была вызвана данная функция.

Значение статуса возврата предопределенной процедуры используется системой при завершении отработки любой предопределенной процедуры, чтобы определить, выполнять ли системно событие, которое вызвало данную предопределенную процедуру.

Замечание: Начальное значение статуса возврата предопределенной процедуры равно 1 (*Выполнить Действие*), которое устанавливается системой при вызове предопределенной процедуры.

В тексте программного модуля возможно использование данного оператора как процедуры или как функции. При использовании в качестве функции, возвращаемое значение соответствует текущему статусу возврата предопределенной процедуры, которое было до вызова данной функции. В данном случае параметр <Статус> можно опускать.

Если параметр <Статус> задан явно, то после выполнения данной функции статус возврата установится в заданное значение. Пример:

* Если в предопределенной процедуре ПриЗаписи установить статус возврата — 0 (например, если неверно заполнены реквизиты), то документ не будет записываться.

```
Процедура ПриЗаписи()
   Если Клиент.Выбран() = 0 Тогда
      Предупреждение("Запись отменена! Не задан клиент!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

* Если в предопределенной процедуре ВводНаОсновании установить статус возврата — 0, то ввод нового документа будет отменен и форма нового документа не откроется.

```
Процедура ВводНаОсновании(ДокОснование)
   Автор = Пользователь;
   ДокВид = ДокОснование.Вид();
   Если (ДокВид = "Счет") ИЛИ (ДокВид = "Счет_фактура")
        ИЛИ (ДокВид = "РасходнаяКредит") ИЛИ (ДокВид = РасходнаяРеализ") Тогда
      Автор = Пользователь;
      ДокВид = ДокОснование.Вид();
      Фирма = ДокОснование.Фирма;
      Клиент = ДокОснование.Клиент;
      Основание = ДокВид + " № " + ДокОснование.НомерДок;
      Сумма = ДокОснование.Итог("Сумма");
      НДС = ДокОснование.Итог("НДС");
   Иначе
      Предупреждение("Недопустимый вид документа основания!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: «Системные предопределенные процедуры»

ОткрытьФорму

Выполняет открытие формы из'встроенного языка.

Синтаксис:

```
ОткрытьФорму (<Параметр1>, <КонтекстФормы>, < Параметр 3>...)
```

Англоязычный синоним:

```
OpenForm
```

Параметры:

<Параметр1>	В зависимости от типа открываемой формы используется различные значения данного параметра. Для журналов, отчетов, списков в конце строки описателя формы может быть указан символ "#" с некоторым идентификатором: "#LLLL". Это используется для того, чтобы данная форма открылась в новом окне, а не активизировала существующее окно этой формы, если оно открыто. Где, LLLL — идентификатор, который позволяет открыть новое окно или активизировать уже открытое с этим идентификатором. Если идентификатор не задан, то обязательно открывается новое окно.
<КонтекстФормы>	Необязательный параметр. Имя переменной, куда можно задать значение любого типа для передачи в открываемую форму. Данное значение будет доступно в открытой форме как атрибут Форма.Параметр. После исполнения данного метода система вернет в данную переменную контекст открытой формы (см. Передача контекста в качестве параметра). С помощью значения этого контекста можно затем произвольно манипулировать открытой формой, пока она открыта. Пока форма открыта, тип значения данного параметра равен 100 (см. ТипЗначения), если закрыта — 0.
<Параметр3>...	В зависимости от типа открываемой формы используется различный состав и значения остальных параметров.

Возвращаемое значение:

Число 1 — если действие выполнено;

Число 0 — если действие не выполнено.

Описание:

Функция ОткрытьФорму позволяет открыть форму документа, справочника, журнала документов, и других форм используемых в системе 1С:Предприятие.

Замечание. Для журналов, отчетов, списков в конце строки описателя формы может быть указан символ "#" с некоторым идентификатором: "#LLLL". Это используется для того, чтобы данная форма открылась в новом окне, а не активизировала существующее окно этой формы, если оно открыто. Где, LLLL — идентификатор, который позволяет открыть новое окно или активизировать уже открытое с этим идентификатором. Если идентификатор не задан, то обязательно открывается новое окно.

Замечание. Метод ОткрытьФорму нельзя вызывать в «теле» глобального модуля (части глобального модуля, расположенной после последней процедуры). Для его вызова при старте программы следует размещать его в процедуре ПриНачалеРаботыСистемы.

Для открытия разных форм следует использовать разный синтаксис вызова данной функции:

Открытие формы существующего документа.

```
ОткрытьФорму(<Документ>, <КонтекстФормы>, <РежимПросмотра>)
```

<Документ>	Значение типа «Документ».
<РежимПросмотра>	Необязательный параметр. Числовое выражение: 1 — открыть форму в режиме только просмотра; 0 — открыть форму в режиме редактирования. -1 (минус единица) в этом случае используется вариант предусмотренный текущим значением параметра «Режим открытия объектов» установленного пользователем. По умолчанию — 0.

Открытие формы нового документа

```
ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <ДокументОснование>)
```

<ОписательОбъекта>	Строковое выражение. "Документ.XXXXX", где XXXXX — идентификатор вида документа;
<ДокументОснование>	Необязательный параметр. Выражение типа «документ», задающий документ-основание для открытия формы ввода нового документа в режиме ввода на основании.

Открытие формы журнала документов.

```
ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)
```

<ОписательОбъекта>	Строковое выражение. "Журнал . XXXXX. YYYYY", где XXXXX — идентификатор журнала документов; YYYYY — идентификатор формы журнала документов; Кроме того, в качестве идентификатора журнала можно указывать идентификатор конкретного вида документа, тогда откроется журнал просмотра документов именно указанного вида, например: ОткрытьФорму("Журнал.РасходнаяНакладная")

Открытие формы журнала подчиненных документов.

```
ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <Документ>)
```

<ОписательОбъекта>	Строковое выражение. "Журнал.Подчиненные";
<Документ>	Значение типа «документ», задающее документ, для которого будут выводиться подчиненные документы.

Открытие формы существующего элемента справочника.

```
ОткрытьФорму(<Элемент>, <КонтекстФормы>, <РежимПросмотра>)
```

<Элемент>	Значение типа «Справочник».
<РежимПросмотра>	Необязательный параметр. Числовое выражение: 1 — открыть форму в режиме только просмотра; 0 — открыть форму в режиме редактирования. -1 (минус единица) в этом случае используется вариант предусмотренный текущим значением параметра «Режим открытия объектов» установленного пользователем. По умолчанию — 0.

Открытие формы нового элемента справочника.

```
ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы> , <ГруппаРодитель>,
                <ФлагГруппы>, <ЭлементВладелец>)
```

<ОписательОбъекта>	Строковое выражение. "Элемент. XXXXX", где XXXXX — вид справочника;
<ГруппаРодитель>	Необязательный параметр. Выражение типа «справочник», задающий родительскую группу для открытия формы ввода нового элемента (группы) справочника.
<ФлагГруппы>	Необязательный параметр. Числовое выражение, задающее форму ввода нового элемента (группы) справочника: 1 — ввод новой группы, 0 — ввод нового элемента. По

умолчанию — 0.

<ЭлементВладелец> Необязательный параметр. Выражение типа «справочник», задающий элемент справочника-владельца для открытия формы ввода нового элемента (группы) подчиненного справочника.

Открытие формы списка справочника.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение.
"Справочник.XXXXX.YYYYY", где XXXXX — вид справочника, YYYYY — имя выбранной формы списка справочника;

Открытие формы отчета.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение. "Отчет. XXXXX" , где XXXXX — вид отчета;

Открытие формы обработки.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение. "Обработка.XXXXX", где XXXXX — вид обработки;

Открытие формы внешнего отчета.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <ИмяФайла>)

<ОписательОбъекта> Строковое выражение. "Отчет";

<ИмяФайла> Строковое выражение — полное имя файла внешнего отчета.

Открытие формы журнала расчетов.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение.
"ЖурналРасчетов. XXXXX", где XXXXX — вид журнала расчетов;

Открытие формы существующего счета.

ОткрытьФорму(<Счет>, <КонтекстФормы>, <РежимПросмотра>)

<Счет> Значение типа «Счет».

<РежимПросмотра> Необязательный параметр. Числовое выражение: 1 — открыть форму в режиме только просмотра; 0 — открыть форму в режиме редактирования. -1 (минус единица) в этом случае используется вариант предусмотренный текущим значением параметра «Режим открытия объектов» установленного пользователем. По умолчанию — 0.

Открытие формы нового счета.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение.
"Счет. XXXXX" , где XXXXX — идентификатор плана счетов;

Открытие формы списка счетов (плана счетов).

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>)

<ОписательОбъекта> Строковое выражение.
"ПланСчетов.XXXX.YYYY", где XXXXX — идентификатор плана счетов; YYYYY — идентификатор формы плана счетов.

Открытие формы существующей операции.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <Документ>, <Проводка>,
<Корреспонденция>, <РежимПросмотра>)

<ОписательОбъекта> Строковое выражение. "Операция".

<Документ> Значение типа «документ», операция которого будет открываться.

<Проводка> Необязательный параметр. Числовое выражение — номер проводки, на которую нужно позиционировать курсор при открытии операции.

<Корреспонденция> Необязательный параметр. Числовое выражение — номер корреспонденции указанной проводки, на которую нужно позиционировать курсор при открытии операции.

<РежимПросмотра> Необязательный параметр. Числовое выражение: 1 — открыть форму в режиме только просмотра; 0 — открыть форму в режиме редактирования. -1 (минус единица) в этом случае используется вариант предусмотренный текущим значением параметра «Режим открытия объектов» установленного пользователем. По умолчанию — 0.

Открытие формы ввода новой операции.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <ТиповаяОперация>)

<ОписательОбъекта> Строковое выражение. "Операция".

<ТиповаяОперация> Необязательный параметр. Строковое выражение — наименование типовой операции, которую следует использовать при вводе новой операции. Если значение не задано или пустое, то ввод операции выполняется без использования типовой.

Открытие формы журнала операций.

ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <Документ>, <Пров>, <Корр>)

<ОписательОбъекта> Строковое выражение.

	"ЖурналОпераций.YYYYY", где YYYYY — идентификатор формы журнала операций;
<Документ>	Значение типа «документ», операция которого будет использована для первоначального позиционирования.
<Пров>	Номер проводки, которая будет использована для первоначального позиционирования курсора, если в журнале операций используется режим показа проводок.
<Корр>	Номер корреспонденции, которая будет использована для первоначального позиционирования курсора, если в журнале операций используется режим показа проводок.

Открытие формы журнала проводок.

```
ОткрытьФорму(<ОписательОбъекта>, <КонтекстФормы>, <Документ>, <Проводка>,
                <Корреспонденция>)
```

<ОписательОбъекта>	Строковое выражение. "ЖурналПроводок.YYYYY", где YYYYY — идентификатор формы журнала проводок;
<Документ>	Значение типа «документ», операция которого будет использована для первоначального позиционирования.
<Проводка>	Необязательный параметр. Числовое выражение — номер проводки операции, на которую нужно в начале позиционироваться.
<Корреспонденция>	Необязательный параметр. Числовое выражение — номер корреспонденции указанной проводки, на которую нужно в начале позиционироваться.

Открытие окна истории значений периодических констант, реквизитов справочников и счетов.

```
ОткрытьФорму("История.Константа.<ИдентКонстанты>", , , <УстНаДату>)
ОткрытьФорму("История.Справочник.<ИдентСправочника>, <ИдентРеквизита>", ,
                <ЭлементСправочника>, <УстНаДату>)
ОткрытьФорму("История.Счет", , <Счет>, <УстНаДату>)
```

<ИдентКонстанты>	Идентификатор периодической константы, как он задан в конфигураторе.
<ИдентСправочника>	Идентификатор справочника, как он задан в конфигураторе.
<ИдентРеквизита>	Идентификатор периодического реквизита справочника, как он задан в конфигураторе.
<ЭлементСправочника>	Элемент справочника, для периодического реквизита которого открывается окно истории значений.
<Счет>	Счет, для которого открывается окно истории значений.
<УстНаДату>	Необязательный параметр. Дата, на которую нужно установить курсор при открытии окна.

Замечание. Допустимо включение в описатель открываемой формы уникального имени открываемой формы через символ "#" с некоторым произвольным идентификатором: "#LLLL". В этом случае открываемая форма закрепляется за выбранным объектом и не происходит обновление формы при смене объекта, например при смене текущей строки списка справочника.

Примеры:

```
Перем Конт;
ОткрытьФорму("История.Справочник.Сотр.Тариф#Ид01", , ТекущийЭлемент());
ОткрытьФорму(Докум, Конт);
ОткрытьФорму("Документ.РасхНакл");
ОткрытьФорму("Журнал.Продажи.КраткаяФорма");
ОткрытьФорму("Отчет", , "C:\1cv7db\extforms\profit.ert");
```

ОткрытьФормуМодально

Выполняет открытие модальной формы из встроенного языка.

Синтаксис:

```
ОткрытьФормуМодально(<Параметр1>, <КонтекстФормы>, <Параметр3>...)
```

Англоязычный синоним:

```
OpenFormModal
```

Параметры: см. параметры метода `ОткрытьФорму`

Возвращаемое значение: см. метод `ОткрытьФорму`.

Описание:

Функция `ОткрытьФормуМодально` позволяет открыть модальную форму документа, справочника, журнала документов, и других форм используемых в системе 1С:Предприятие. Синтаксис и описание см. метод `ОткрытьФорму`.

См. также: `ОткрытьФорму`

ТипЗначения

Получить тип значения данных.

Синтаксис:
`ТипЗначения(<Значение>)`
Англоязычный синоним:
`ValueType`
Параметры:
`<Значение>` Выражение, тип данных значения которого надо определить.
Возвращаемое значение:
Функция возвращает числовое значение:
0 — неопределенный тип данных;
1 — числовой тип данных;
2 — строковый тип данных;
3 — тип данных — дата;
10 — агрегатный тип данных «*Перечисление*»;
11 — агрегатный тип данных «*Справочник*»;
12 — агрегатный тип данных «*Документ*»;
13 — агрегатный тип данных «*Календарь*»;
14 — агрегатный тип данных «*ВидРасчета*»;
15 — агрегатный тип данных «*Счет*»;
16 — агрегатный тип данных «*ВидСубконто*»;
17 — агрегатный тип данных «*ПланСчетов*»;
100 — так называемый внешний объект. В этот класс попадают все агрегатные объекты, не вошедшие в вышеперечисленный список, такие как «*Текст*», «*Таблица*», «*Запрос*», «*ЖурналРасчетов*» и т. п.
Описание:
Функция `ТипЗначения` определяет, к какому типу данных принадлежит переданный параметр `<Значение>` и возвращает числовой результат:
Пример:
```
Если ТипЗначения(Код)<>1 Тогда
   Предупреждение("Код имеет не числовой тип !!!");
КонецЕсли;
```

ТипЗначенияСтр

Получить строковое обозначение типа данных.
Синтаксис:
`ТипЗначенияСтр(<Значение>)`
Англоязычный синоним:
`ValueTypeStr`
Параметры:
`<Значение>` Выражение, тип данных значения которого надо определить.
Возвращаемое значение:
Функция возвращает строковое значение в русскоязычном или в англоязычном написании — в зависимости от текущей установки основного языка конфигурации:

Русскоязычное	Англоязычное	Описание
Неизвестный Объект	UnknownObject	неопределенный тип данных;
Число	Number	числовой тип данных;
Строка	String	строковый тип данных;
Дата	Date	тип данных — дата;
Перечисление	Enum	агрегатный тип данных «*Перечисление*»;
Справочник	Reference	агрегатный тип данных «*Справочник*»;
Документ	Document	агрегатный тип данных «*Документ*»;
Регистр	Register	агрегатный тип данных «*Регистр*»;
Календарь	Calendar	агрегатный тип данных «*Календарь*»;
ВидРасчета	Calculation	агрегатный тип данных «*ВидРасчета*»;
ЖурналРасчетов	CalcJournal	агрегатный тип данных «*ЖурналРасчетов*»;
ПланСчетов	ChartOfAccounts	агрегатный тип данных «*ПланСчетов*»;
Счет	Account	агрегатный тип данных «*Счет*»,
Операция	Operation	агрегатный тип данных «*Операция*»;
КорректныеПроводки	CorrectEntries	агрегатный тип данных «*КорректныеПроводки*»;
БухгалтерскиеИтоги	BookkeepingTotals	агрегатный тип данных «*БухгалтерскиеИтоги*»;
Таблица	Table	агрегатный тип данных «*Таблица*»;
Текст	Text	агрегатный тип данных «*Текст*»;

Запрос	Query	агрегатный тип данных «*Запрос*»;
СписокЗначений	ValueList	агрегатный тип данных «*СписокЗначений*»;
ТаблицаЗначений	ValueTable	агрегатный тип данных «*ТаблицаЗначений*»;
Периодический	Per1odic	агрегатный тип данных «*Периодический*»;
Картинка	Picture	агрегатный тип данных «*Картинка*»;
ГрупповойКонтекст	GroupContext	локальный контекст программного модуля, передаваемый при помощи ключевого слова `Контекст` или возвращаемый методом `ОткрытьПодбор`;
OLE	OLE	OLE-объект;

Описание:

Функция `ТипЗначенияСтр` определяет, к какому типу данных принадлежит переданный параметр `<Значение>` и возвращает соответствующе строковое значение. Название агрегатного типа данных передается либо в русском либо в английском написании — в зависимости от текущей установки основного языка конфигурации.

Пример:

```
Предупреждение("Код имеет тип - " + ТипЗначенияСтр(Код));
```

См. также: `ОсновнойЯзык`

ПустоеЗначение

Проверить значение любого типа.

Синтаксис:

```
ПустоеЗначение(<Значение>)
```

Англоязычный синоним:

```
Empty Value
```

Параметры:

`<Значение>` Выражение любого типа данных, значение которого проверяется на «пустое».

Возвращаемое значение:

Функция возвращает числовое значение: 1 — значение пустое; 0 — значение не пустое.

Описание:

Функция `ПустоеЗначение` определяет, является ли пустым переданное в параметре значение. При этом применяется следующее правило:

- значение неопределенного типа — всегда пустое;
- значение типа «строка» проверяется как в методе `ПустаяСтрока`;
- значение типа «число» проверяется на равенство нулю;
- значение типа «дата» проверяется на пустое значение;
- значения следующих типов: «справочник», «перечисление», «документ», «счет», «вид субконто», «план счетов», «календарь» проверяются как в методах Выбран для соответствующего типа объекта.

Пример:

```
Если ПустоеЗначение(Код) = 1 Тогда
   Предупреждение("Задайте непустое значение Кода !!!");
КонецЕсли;
```

ПолучитьПустоеЗначение

Получить пустое значение заданного типа данных.

Синтаксис:

```
ПолучитьПустоеЗначение(<Тип>)
```

Англоязычный синоним:

```
GetEmptyValue
```

Параметры:

`<Тип>` Необязательный параметр. Строка или вид субконто или объект метаданных, задающий тип данных. Если параметр не задан, то функция возвращает пустое значение неопределенного типа.

Возвращаемое значение:

Функция `ПолучитьПустоеЗначение` возвращает пустое значение заданного типа данных.

Описание:

Функция возвращает пустое значение заданного типа данных.

Пример:

```
Если ВыбФирма = ПолучитьПустоеЗначение("Справочник.Фирмы") = 1 Тогда
   Предупреждение("Задайте непустое значение Фирмы");
КонецЕсли;
```

НазначитьВид

Для значений типа «Документ неопределенного вида», «Справочник неопределенного вида», «Счет неопределенного вида» назначает конкретный вид.

Синтаксис:

```
НазначитьВид(<Значение>, <Вид>)
```

Англоязычный синоним:

```
SetKind
```

Параметры:

<Значение>	Значение типа документ/справочник/счет неопределенного вида — обычно реквизит документа, справочника или диалога формы.
<Вид>	Строка — вид значения может быть задан строкой, содержащей идентификатор конкретного справочника, документа, плана счетов. Вид субконто — вид значения может быть задан видом субконто, который имеет тип соответственно конкретного справочника, документа, счета.

Описание:

Процедура НазначитьВид используется для установки значениям типа «Документ неопределенного вида», «Справочник неопределенного вида», «Счет неопределенного вида» конкретный вид. При этом, если существующее значение не соответсвует вновь установленному виду — то оно очищается.

Данная процедура используется обычно для значений реквизитов в диалогах форм. В качестве первого параметра передается значение типа Справочник, Документ или Счет, в котором нужно установить вид. Во втором параметре указывается вид в качестве строки — идентификатора или вида субконто.

Данная процедура может использоваться в сочетании с методом элемента формы НеИзменятьВид. Это позволяет регулировать программно собственно вид, а само значение предоставить выбирать пользователю интерактивно.

Пример:

* Например, реквизиту документа «Контрагент» типа справочник неопределенного вида, в зависимости от значений других реквизитов можно установить вид «Организации» или «Сотрудники».

```
Процедура ВводНаОсновании(ДокОсн)
   Если ДокОсн.Вид() = "Счет" Тогда
      НазначитьВид(Контрагент, «Организации»);
   Иначе
      НазначитВид(Контрагент, «Сотрудники»);
   КонецЕсли;
   Форма.Контрагент.НеИзменятьВид(1);
КонецПроцедуры
```

ЗаписьЖурналаРегистрации

Выводит строку текста в системный журнал регистрации.

Синтаксис:

```
ЗаписьЖурналаРегистрации(<Коммент>, <ТипСобытия>, <Событие>, <Объект>,
                         <Категория>)
```

Англоязычный синоним:

```
LogMessageWrite
```

Параметры:

<Коммент>	Строковое выражение, комментарий к событию. Если этот параметр — единственный переданный при вызове метода, то тогда в журнал записывается стандартное дополнительное событие с указанным комментарием.
<ТипСобытия>	Необязательный параметр. Строковое выражение -тип события. По умолчанию или при пустой строке «Дополнительные события».
<Событие>	Необязательный параметр. Строковое выражение -событие. По умолчанию «Дополнительное событие».
<Объект>	Необязательный параметр. Объект события, по умолчанию отсутствует. Для переданных в функцию объектов типа Документ/ Справочник/ Счет представление объекта будет записано автоматически (это не зависит от типа события и самого события).
<Категория>	Необязательный параметр. Число — категория события: 1 — администрирование; 2 — изменение данных; 3 — информация; 4 — предупреждение; 5 — ошибка. Значение по умолчанию 3.

Описание:

Процедура ЗаписьЖурналаРегистрации позволяет эмулировать запись в системный журнал регистрации информации о возникновении важных системных событий.

Системой зарезервированы следующие типы событий/события:

Документ / Открыт	Справочник/СнятаПометкаНаУдаление
Документ / Новый	
Документ / Удален	Константа / ЗначениеЗаписано
Документ / Записан	Константа / ЗначениеУдалено
Документ / НаЗаписан	
Документ / Проведен	ЖурналРасчетов / ИзмененРасчетныйПериод
Документ / ПроведенЗаднимЧислом	ЖурналРасчетов / ОткатНазадРасчетногоПериода
Документ / НеПроведен	ЖурналРасчетов / НеудачнаяПопыткаИзмененияПериода
Документ / СделанНеПроведенным	ЖурналРасчетов / ОтменаРучногоРедактированияРезультата
Документ / ЗаписанИПроведенЗаднимЧислом	ЖурналРасчетов / ИзмененРезультат
Документ / ЗаписанИПроведен	
Документ / ПомеченНаУдаление	Счет / ЗаписанНовый
Документ / СнятаПометкаНаУдаление	Счет / Записан
Документ / Изменена ОперацияДокумента	Счет / ЗначениеРеквизитаЗаписано
Документ / ПроводкиВключены	Счет / ЗмачениеРеквизитаУдалено
Документ / ПроводкиВыключены	Счет / Удален
Документ / ИзмененоВремя	Счет / ПомеченНаУдаление
Справочник / ЗаписанНовый	Счет / СнятаПометкаНаУдаление
Справочник / Записан	
Справочник / Удален	КорректнаяПроводка / ЗаписанаНовая
Справочник / ЗначениеРеквизитаЗаписано	КорректнаяПроводка / Записана
Справочник / ЗначениеРеквизитаУдалено	КорректнаяПроводка / Удалена
Справочник / ПеренесенВДругуюГруппу	
Справочник / ПомеченНаУдаление	ОбщиеСобытия / ИзменениеТА

Незарезервированные имена позволяют пользователю фактически указывать свои собственные типы событий и события. Это значит, что если в параметре <ТипСобытия> записать "Отчет" а в параметре <Событие> "Сформирован", то при просмотре журнала регистрации можно будет фильтровать по событию «Сформирован» типа событий «Отчет».

Пример:

```
ЗаписьЖурналаРегистрации("Доброе утро!")
```

ПрефиксАвтоНумерации

Установить префикс для автоматического создания новых номеров.

Синтаксис:

```
ПрефиксАвтоНумерации(<ИмяВида>, <Префикс>)
```

Англоязычный синоним:

```
AutoNumPrefix
```

Параметры:

<ИмяВида>	Строковое выражение с полным названием справочника или документа, как он назван в конфигурации.
<Префикс>	Строковое выражение — префикс номеров документов или кодов элементов справочника.

Описание:

Процедура ПрефиксАвтоНумерации устанавливает префикс для автоматического создания новых номеров документов или кодов элементов справочника.

Вызов данного метода может быть использован для установки префикса всем документам или справочникам. При этом используется символ "*" вместо идентификатора вида документа/справочника, например

```
ПрефиксАвтоНумерации("Документ.*", "ПР-")
```

Пример:

```
ПрефиксАвтоНумерации("Документ.РасходнаяНакл", "Рн-");
```

См. также: ПрефиксНомера, ПрефиксКода, УстановитьНовыйНомер, УстановитьНовыйКод.

ПолучитьЗначенияОтбора

Выбирать все существующие значения отбора.

Синтаксис:

```
ПолучитьЗначенияОтбора(<ИмяОтбора>, <СписокЗначений> , [<ДатаНач>], [<ДатаКон>])
```

Англоязычный синоним:

```
GetSelect ionValues
```

Параметры:

<ИмяОтбора>	Строковое выражение с полным названием общего реквизита документа или графы

	отбора, как он назван в конфигурации.
<СписокЗначений>	Идентификатор переменной. Если при вызове метода данная переменная содержит объект типа «СписокЗначений», то система заполнит его всеми возможными существующими значениями отбора. Если при вызове метода данная переменная содержит любое другое значение кроме объекта типа «СписокЗначений», то система сама создаст объект типа «СписокЗначений», заполнит его существующими значениями отбора и присвоит переменной ссылку на этот объект.
<ДатаНач>	Необязательный параметр. Дата начала интервала в котором проводить отбор. Если параметр не задан, то отбор будет производиться по всем данным.
<ДатаКон>	Необязательный параметр. Дата конца интервала в котором проводить отбор. Если параметр не задан или равен 0, то отбор будет производиться до ТА.

Возвращаемое значение:

Числовое значение: 1 — операция выполнена успешно; 0 — операция не выполнена.

Описание:

Функция ПолучитьЗначенияОтбора выбирает все возможные существующие значения отбора <ИмяОтбора>, заполняет выбранными значениями объект типа «СписокЗначений», и передает ссылку на этот объект в переменной <СписокЗначений>.

Замечание. Функция ПолучитьЗначенияОтбора может работать и с выбранными в конфигурации отборами операций и проводок ("СуммаОперации", "Содержание", "Сумма", "Валюта", "Количество", "ВалСумма", "Счет", "СчетДт", "СчетКт", "ПланСчетов", реквизитам операции, реквизитам проводки, видам субконто) для которых установлен режим отбора.

Пример:

```
// Выбрать существующие значения Авторов документов
ПолучитьЗначенияОтбора("Автор", Спис, , );
```

См. также: «Работа со Списком Значений»

КомандаСистемы

Вызывает на исполнение команду DOS.

Синтаксис:

```
КомандаСистемы(<СтрокаКоманды>)
```

Англоязычный синоним:

```
System
```

Параметры:

<СтрокаКоманды>	Строковое выражение, содержащее строку команды.

Описание:

Процедура КомандаСистемы вызывает на исполнение команду, как если бы она была введена в командной строке DOS. При выполнении команды загружается исполняемый файл как «верхняя», активная программа.

Если загружается Windows-приложение, то управление немедленно возвращается в систему 1С:Предприятие на следующий за оператором КомандаСистемы оператор и система продолжает выполняться как фоновая прикладная программа. Пользователь может возвратиться в запускающую программу или закончив выполнение запущенной, или переключившись обратно посредством списка задач Windows.

Если в параметре <СтрокаКоманды> не содержится путь к загружаемой программе в файловой системе к программе, то поиск происходит в следующей последовательности:

- текущий системный каталог;
- каталог Windows;
- системный каталог Windows;
- все каталоги, указанные в переменной PATH;

Пример:

* В данном примере запускается редактор текста notepad.exe и в него загружается файл textl. txt.

```
КомандаСистемы("notepad textl.txt");
```

ЗапуститьПриложение

Выполняет запуск внешнего приложения.

Синтаксис:

```
ЗапуститьПриложение(<СтрокаКоманды>)
```

Англоязычный синоним:

```
RunApp
```

Параметры:

<СтрокаКоманды>	Строковое выражение, содержащее строку команды запуска внешнего приложения или имя файла, чтобы открыть его с помощью ассоциированного для данного вида файлов приложения.

Описание:

Процедура ЗапуститьПриложение вызывает запуск внешнего приложения. В отличие от функции КомандаСистемы запуск приложения выполняется непосредственно, минуя командный интерпретатор. При выполнении команды исполняемый файл загружается как «верхняя», активная программа.

Процедура, кроме того, может открыть файл с помощью ассоциированного для данного вида файлов приложения. (При условии, что таковое установлено на данном компьютере).

Если загружается Windows-приложение, то управление немедленно возвращается в систему 1С:Предприятие на следующий за оператором ЗапуститьПриложение оператор и система продолжает выполняться как фоновая прикладная программа. Пользователь может возвратиться в запускающую программу или закончив выполнение запущенной, или переключившись обратно посредством списка задач Windows.

Если в параметре <СтрокаКоманды> не содержится путь к загружаемой программе в файловой системе к программе, то поиск происходит в следующей последовательности:

- текущий системный каталог;
- каталог Windows;
- системный каталог Windows;
- все каталоги, указанные в переменной PATH;

Пример:

* В данном примере запускается редактор текста notepad.exe и в него загружается файл textl. txt.

```
ЗапуститьПриложение("notepad textl.txt");
```

ЗавершитьРаботуСистемы

Выполняет запуск внешнего приложения.

Синтаксис:

```
ЗавершитьРаботуСистемы(<ФлагСохранения>)
```

Англоязычный синоним:

```
ExitSystem
```

Параметры:

<ФлагСохранения>	Необязательный параметр. Флаг запроса сохранения открытых объектов. Если 0, то не запрашивается сохранение не записанных документов, элементов справочников и т. п. Если 1, то сохранение запрашивается. Значение по умолчанию 1.

Описание:

Процедура ЗавершитьРаботуСистемы вызывает завершение работы системы.

Пример:

```
ЗавершитьРаботуСистемы(0);
```

НайтиПомеченныеНаУдаление

Находит все помеченные на удаление объекты.

Синтаксис:

```
НайтиПомеченныеНаУдаление(<Объекты>)
```

Англоязычный синоним:

```
FindMarkedForDelete
```

Параметры:

<Объекты>	Идентификатор объекта типа «СписокЗначений», в который данная процедура помещает найденные объекты.

Описание:

Процедура НайтиПомеченныеНаУдаление находит все помеченные на удаление объекты и помещает их в список значений.

Пример:

```
Список = СоздатьОбъект("Список Значений");
НайтиПомеченныеНаУдаление(Список);
```

НайтиСсылки

Находит ссылки на объекты, переданные в списке значений.

Синтаксис:

```
НайтиСсылки(<Объекты>, <Ссылки>)
```

Англоязычный синоним:

```
FindReferences
```

Параметры:

<Объекты>	Конкретное значение объекта или объект типа «СписокЗначений», в котором данной процедуре передаются объекты, по которым надо найти ссылки.
<Ссылки>	Идентификатор объекта типа «ТаблицаЗначений», в который данная процедура помещает найденные ссылки на объекты. Таблица значений состоит из 3 колонок: 1. Объект — объект, на который ссылаются; 2. Ссылка — объект, который содержит ссылку; 3. Комментарий — комментарий к ссылке.

Описание:

Процедура НайтиСсылки находит ссылки на переданные в списке объекты, и помещает их в таблицу значений.

Пример:

```
Список = СоздатьОбъект("СписокЗначений");
ТаблицаССылок = СоздатьОбъект("ТаблицаЗначений");
НайтиПомеченныеНаУдаление(Список);
НайтиСсылки(Список, ТаблицаССылок);
```

УдалитьОбъекты

Удалить объекты, переданные в списке значений.

Синтаксис:

```
УдалитьОбъекты(<Объекты>, <Проверять>, <Ссылки>)
```

Англоязычный синоним:

```
DeleteObjects
```

Параметры:

<Объекты>	Конкретное значение объекта или объект типа «СписокЗначений», в котором данной процедуре передаются объекты, которые надо удалить.
<Проверять>	Необязательный параметр. Число: 1 — перед удалением проверяется — нет ли ссылок на удаляемый объект. Если есть, то объект не удаляется, а ссылки помещаются в таблицу значений <Ссылки>. Любое другое число — проверку не выполнять. Значение по умолчанию: 1.
<Ссылки>	Необязательный параметр. Идентификатор объекта типа «ТаблицаЗначений», в который данная процедура помещает найденные ссылки на объекты. Таблица значений состоит из 3 колонок: 1. Объект — объект, на который ссылаются; 2. Ссылка — объект, который содержит ссылку; 3. Комментарий — комментарий к ссылке.

Описание:

Процедура УдалитьОбъекты удаляет объекты, переданные в списке значений.

Пример:

```
Список = СоздатьОбъект("СписокЗначений");
ТаблицаССылок = СоздатьОбъект("ТаблицаЗначений");
НайтиПомеченныеНаУдаление(Список);
УдалитьОбъекты(Список, 1, ТаблицаССылок);
```

ОбработкаОжидания

Инициирует периодический вызов процедуры глобального модуля с заданным интервалом времени.

Синтаксис:

```
ОбработкаОжидания([<ИмяПроцедуры>], [<ИнтервалВызова>])
```

Англоязычный синоним:

```
IdleProcessing
```

Параметры:

<ИмяПроцедуры>	Необязательный параметр. Строковое выражение — имя процедуры глобального модуля, которая будет вызываться периодически с интервалом времени <ИнтервалВызова>. Тело процедуры <ИмяПроцедуры> должно быть написано разработчиком конфигурации в глобальном программном модуле. Если в качестве параметра передается «пустая строка», то ранее запущенный процесс прекращается. Если параметр опущен, то данная процедура ОбработкаОжидания просто возвращает имя процедуры глобального модуля, которая назначена для периодического запуска в текущий момент.
<ИнтервалВызова>	Необязательный параметр. Числовое выражение — интервал времени в секундах, с которым периодически будет вызываться процедура глобального модуля <ИмяПроце-

дуры>.
Если в качестве параметра передается 0 (ноль), то ранее запущенный процесс прекращается.
Если параметр опущен, то данная процедура ОбработкаОжидания просто возвращает имя процедуры глобального модуля, которая назначена для периодического запуска в текущий момент.

Возвращаемое значение:

Имя процедуры глобального модуля, которая назначена для периодического запуска (на момент до исполнения процедуры).

Описание:

Функция ОбработкаОжидания инициирует периодический вызов процедуры глобального модуля <ИмяПроцедуры> с интервалом времени <ИнтервалВызова>.

Пример:

```
Процедура СформироватьТекущийОтчетПродаж
КонецПроцедуры;
ОбработкаОжидания("СформироватьТекущийОтчетПродаж", 60);
```

Процедуры и функции обработки значений

ЗначениеВСтрокуВнутр

Преобразование значения объекта агрегатного типа из внутреннего представления в строковое.

Синтаксис:

```
ЗначениеВСтрокуВнутр(<Выражение>)
```

Англоязычный синоним:

```
ValueToStringInternal
```

Параметры:

<Выражение>	Выражение, вычисленное значение которого следует перевести в строковое представление. В качестве параметра для данного метода допускается передавать значения любых типов данных, доступных в конфигурации, как базовых, так и агрегатных типов данных, за исключением тех типов данных, которые не имеют собственно значения — это касается таких объектов как «Периодический», «Таблица», «ФС», «Запрос», «XBase» и т. п. Однако, объекты типа «СписокЗначений» и «ТаблицаЗначений» использовать можно — они преобразуются в строковое представление.

Возвращаемое значение:

Результирующее строковое значение.

Описание:

Применять функцию ЗначениеВСтрокуВнутр имеет смысл только в особых случаях. Например, если в рамках одной конфигурации требуется несколько объектов упаковать в одну строку. При помощи функции ЗначениеВСтрокуВнутр объекты можно преобразовать в строковые значения, которые затем можно объединить в единую строку операцией конкатенации. В дальнейшем эту строку можно будет разобрать на составляющие и сделать обратное преобразование при помощи функции ЗначениеИзСтрокиВнутр.

Замечание: Строковое представление агрегатного типа данных имеет специальный системный формат, т. к. предназначено не для отображения, а просто для возможности выполнения строковых операций с объектами (в рамках одной конфигурации), например при упаковке нескольких объектов в одну строку. Данный системный формат использует внутренний системный идентификатор данных, поэтому этот формат предназначен для работы в рамках единой информационной базы. При распаковке такого строкового представления функция ЗначениеИзСтрокиВнутр восстанавливает значение объекта по его системному идентификатору.

Пример:

* Допустим, в конфигурации формируется отчет с использованием объекта «Таблица». Для обеспечения возможности получать дополнительные сведения по отчету, в таблицах есть возможность в каждой ячейке хранить вычисляемое значение, которое можно затем использовать для обработки. Однако, в ячейке таблицы можно хранить только одно значение. В случае, если все же для обработки необходимо иметь несколько значений, то их можно с помощью метода ЗначениеВСтрокуВнутр упаковать в одну строку. Например, в ячейке таблицы запишем формулу значения ячейки в следующем виде:

```
ЗначениеВСтрокуВнутр(Запрос.Клиент) + РазделительСтрок +
                    ЗначениеВСтрокуВнутр(Запрос.Товар)
```

Процедура обработки значения ячейки должна в себе содержать алгоритм распаковки значений с использованием метода ЗначениеИзСтрокиВнутр.

```
Процедура ОбработкаЯчейкиТаблицы(Значение, ФлагСтандартнойОбработки)
```

```
   Если ТипЗначенияСтр(Значение) = "Строка" Тогда
      Орг = ЗначениеИзСтрокиВнутр(СтрПолучитьСтроку(Значение, 1));
      Тов = ЗначениеИзСтрокиВнутр(СтрПолучитьСтроку(Значение, 2));
      Если Тов.Выбран() = 0 Тогда
         Возврат;
      КонецЕсли;
      Карт(Орг, Тов);
      ФлагСтандартнойОбработки = 0;
      Возврат;
   КонецЕсли;
   ФлагСтандартнойОбработки = 1;
КонецПроцедуры
```

См. также: ЗначениеИзСтрокиВнутр, ЗначениеТекущейЯчейки, ОбработкаЯчейкиТаблицы

ЗначениеИзСтрокиВнутр

Преобразование значения объекта агрегатного типа из строкового системного представления во внутреннее.

Синтаксис:

ЗначениеИзСтрокиВнутр(<СистСтрока>)

Англоязычный синоним:

ValueFromStringInternal

Параметры:

<СистСтрока>	Строковое выражение, представление объекта агрегатного типа данных в системном строковом виде.

Возвращаемое значение:

Объект агрегатного типа данных, который получен из строкового системного представления.

Описание:

Применять функцию ЗначениеИзСтрокиВнутр имеет смысл только в особых случаях. Например, если требуется несколько объектов упаковать в одну строку. При помощи функции ЗначениеВСтрокуВнутр объекты можно преобразовать в строковые значения, которые затем можно объединить в единую строку операцией конкатенации. В дальнейшем эту строку можно будет разобрать на составляющие и сделать обратное преобразование при помощи функции ЗначениеИзСтрокиВнутр.

Замечание: Строковое представление агрегатного типа данных имеет специальный системный формат, т. к. предназначено не для отображения, а просто для возможности выполнения строковых операций с объектами (в рамках одной конфигурации), например при упаковке нескольких объектов в одну строку. Данный системный формат использует внутренний системный идентификатор данных, поэтому этот формат предназначен для работы в рамках единой информационной базы. При распаковке такого строкового представления функция ЗначениеИзСтрокиВнутр восстанавливает значение объекта по его системному идентификатору.

Замечание. При использовании метода ЗначениеИзСтрокиВнутр() для значения типа «СписокЗначений», являющимся реквизитом формы, его идентификатор следует передавать в качестве второго параметра. В этом случае метод следует использовать как процедуру, т. е. не использовать возвращаемое значение. Пример:

```
ВосстановитьЗначение("СистСтрокаМойСпис", ИдентМойСпис);
```

Пример: См. предыдущий пример

См. также: ЗначениеВСтрокуВнутр, ЗначениеТекущейЯчейки

ЗначениеВСтроку

Преобразование значения из внутреннего представления в строковое.

Синтаксис:

ЗначениеВСтроку(<Объект>)

Англоязычный синоним:

ValueToString

Параметры:

<Объект>	Значение, которое следует перевести в строковое представление. В качестве параметра для данного метода допускается передавать значения любых типов данных, доступных в конфигурации, как базовых, так и агрегатных типов данных, за исключением тех типов данных, которые не имеют собственно значения — это касается таких объектов как «Периодический», «Таблица», «ФС», «Запрос», «XBase» и т. п. Однако, объекты типа «СписокЗначений» и «ТаблицаЗначений» использовать можно — он преобразуется в строковое пред-

ставление.

Возвращаемое значение:

Строковое представление переданного значения.

Описание:

Применять функцию ЗначениеВСтроку имеет смысл только в особых случаях. Например, если требуется несколько объектов упаковать в одну строку. При помощи функции ЗначениеВСтроку объекты можно преобразовать в строковые значения, которые затем можно объединить в единую строку операцией конкатенации. В дальнейшем эту строку можно будет разобрать на составляющие и сделать обратное преобразование при помощи функции ЗначениеИзСтроки.

Замечание: Строковое представление имеет специальный системный формат, т. е. предназначено не для отображения, а просто для возможности выполнения строковых операций с объектами, например при экспорте-импорте информации между разными конфигурациями (например, при помощи OLE). При преобразовании агрегатных типов данных, данный системный формат использует идентификаторы объектов, как они даны в конфигураторе, а также значения кода (для элементов справочников) и номера (для документов), поэтому при передаче такого строкового представления в другую (похожую) конфигурацию функция ЗначениеИзСтроки сделает попытку восстановить значение объекта по его строковому описанию.

Пример:

```
СтрДок = ЗначениеВСтроку(ВыбДокумент);
```

См. также: ЗначениеИзСтроки

ЗначениеИзСтроки

Преобразование из строкового системного представления во внутреннее значение объекта.

Синтаксис:

```
ЗначениеИзСтроки(<СистСтрока>)
```

Англоязычный синоним:

```
ValueFromString
```

Параметры:

<СистСтрока>	Строковое выражение, содержащее представление объекта, полученное ранее при помощи метода ЗначениеВСтроку.

Возвращаемое значение:

Значение, которое получено из строкового системного представления.

Описание:

Применять функцию ЗначениеИзСтроки имеет смысл только в особых случаях. Например, если требуется несколько объектов упаковать в одну строку. При помощи функции ЗначениеВСтроку объекты можно преобразовать в строковые значения, которые затем можно объединить в единую строку операцией конкатенации. В дальнейшем эту строку можно будет разобрать на составляющие и сделать обратное преобразование при помощи функции ЗначениеИзСтроки.

Замечание: Строковое представление агрегатного типа данных имеет специальный системный формат, т. е. предназначено не для отображения, а просто для возможности выполнения строковых операций с объектами, например при экспорте-импорте информации между разными конфигурациями (например, при помощи OLE). При преобразовании агрегатных типов данных, данный системный формат использует идентификаторы объектов, как они даны в конфигураторе, а также значения кода (для элементов справочников) и номера (для документов), поэтому при передаче такого строкового представления в другую (похожую) конфигурацию функция ЗначениеИзСтроки сделает попытку восстановить значение объекта по его строковому описанию.

Пример:

```
Док = ЗначениеИзСтроки(СтрДок);
```

См. также: ЗначениеВСтроку

ЗначениеВФайл

Сохраняет значение в файле.

Синтаксис:

```
ЗначениеВФайл(<ИмяФайла>, <Значение>, <Формат>)
```

Англоязычный синоним:

```
ValueToFile
```

Параметры:

<ИмяФаила>	Строковое выражение, задающее имя файла.
<Значение>	Значение, которое следует сохранить в файле. В качестве параметра для данного метода допускается передавать значения любых типов данных, доступных в конфигурации, как

	базовых, так и агрегатных типов данных, за исключением тех типов данных, которые не имеют собственно значения — это касается таких объектов как «Периодический», «Таблица», «ФС», «Запрос», «XBase» и т. п. Однако, объекты типа «СписокЗначений» и «ТаблицаЗначе-ний» использовать можно.
<Формат>	Необязательный параметр. Число: 1 — сохранение во внутреннем формате; иначе во внешнем. Значение по умолчанию: 1.

Возвращаемое значение:
Число: 1 — функция выполнена успешно; 0 — функция не выполнена.
Описание:
Функция ЗначениеВФайл позволяет сохранить любое значение в файле. В дальнейшем этот файл можно будет распаковать и сделать обратное преобразование при помощи функции ЗначениеИзФайла.
Пример:
```
Результат = ЗначениеВФайл(ВыбИмяФайла, ВыбДокумент);
```
См. также: ЗначениеИзФайла

ЗначениеИзФайла

Восстанавливает значение из файла.
Синтаксис:
ЗначениеИзФайла(<ИмяФайла>, <Значение>, <Формат>)
Англоязычный синоним:
ValueFromFile
Параметры:

<ИмяФайла>	Строковое выраженние, задающее имя файла.
<Значение>	Необязательный параметр. Идентификатор переменной, куда система вернет значение, которое получено из файла.
<Формат>	Необязательный параметр. Число: 1 — сохранение во внутреннем формате; иначе во внешнем. Значение по умолчанию: 1.

Возвращаемое значение:
Значение, которое получено из файла.
Описание:
Функция ЗначениеИзФайла восстанавливает значение из файла, которое упаковано туда при помощи функции ЗначениеВФайл.
Пример:
```
Док = ЗначениеИзФайла(ВыбИмяФайла);
```
См. также: ЗначениеВФайл

СохранитьЗначение

Сохранить для пользователя некоторое значение между сеансами.
Синтаксис:
СохранитьЗначение(<Идентификатор>, <Значение>)
Англоязычный синоним:
SaveValue
Параметры:

<Идентификатор>	Строковое выражение, наименование сохраняемого значения. Данное наименование используется для идентификации сохраняемых значений.
<Значение>	Выражение любого типа. Объекты типа «Список-Значений» и «ТаблицаЗначений» использовать можно.

Описание:
Применять процедуру СохранитьЗначение следует, например, если требуется запомнить некоторые значения до следующего запуска системы.
Например, можно сохранять для данного пользователя значение того склада, с которого он обычно выписывает накладные. При запуске системы в следующий раз данное значение может использоваться как значение склада по умолчанию для данного пользователя.
Пример:
```
СохранитьЗначение("МойСклад", Склад);
```
См. также: ВосстановитьЗначение

ВосстановитьЗначение

Восстановить сохраненное для пользователя значение.
Синтаксис:

ВосстановитьЗначение(<Идентификатор>)

Англоязычный синоним:

RestoreValue

Параметры:

<Идентификатор>	Строковое выражение, наименование восстанавливаемого значения. Данное наименование используется для идентификации сохраненных значений.

Возвращаемое значение:

Значение, сохраненное для пользователя.

Описание:

Применять функцию ВосстановитьЗначение следует, например, если требуется восстановить ранее запомненное (в предыдущем сеансе работы) некоторые значение системы.

Например, можно сохранять для данного пользователя значение того склада, с которого он обычно выписывает накладные. При запуске системы в следующий раз данное значение может использоваться как значение склада по умолчанию для данного пользователя.

Замечание. При использовании метода ВосстановитьЗначение() для значения типа «СписокЗначений», являющимся реквизитом формы, его идентификатор следует передавать в качестве второго параметра. В этом случае метод следует использовать как процедуру, т. е. не использовать возвращаемое значение. ВосстановитьЗначение("СистСтрокаМойСпис", ИдентМойСпис);

Пример:

```
ДефСклад = ВосстановитьЗначение("МойСклад");
```

См. также: СохранитьЗначение

Процедуры и функции компоненты «Оперативный учет»

ПолучитьТА

Получить дату и время *Точки Актуальности* итогов в текстовом виде. *Данная функция используется только при наличии компоненты. «Оперативный учет».*

Синтаксис:

ПолучитьТА()

Англоязычный синоним:

GetAP

Возвращаемое значение:

Строковое значение, содержащее дату и время ТА.

Описание:

Функция ПолучитьТА возвращает дату и время *Точки Актуальности* итогов в текстовом виде.

Пример:

```
Дата_И_Время_ТА = ПолучитьТА();
```

ПолучитьДатуТА

Получить дату *Точки Актуальности* итогов. *Данная функция используется только при наличии компоненты «Оперативный учет».*

Синтаксис:

ПолучитьДатуТА()

Англоязычный синоним:

GetDateOfAP

Возвращаемое значение:

Значение типа «дата», содержащее дату ТА.

Описание:

Функция ПолучитьДатуТА возвращает значение (типа «дата») даты *Точки Актуальности* итогов.

Пример:

```
Дата_ТА = ПолучитьДатуТА();
```

ПолучитьВремяТА

Получить значение времени *Точки Актуальности* итогов. *Данная функция используется только при наличии компоненты «Оперативный учет».*

Синтаксис:

ПолучитьВремяТА(<Часы>, <Минуты>, <Секунды>)

Англоязычный синоним:

GetTimeOfAP

Параметры:

<Часы>	Идентификатор переменной, в которую метод возвращает строковое значение часа ТА.
<Минуты>	Идентификатор переменной, в которую метод возвращает строковое значение минут ТА.
<Секунды>	Идентификатор переменной, в которую метод возвращает строковое значение секунд ТА.

Возвращаемое значение:

Строковое значение времени ТА в виде " ЧЧ. ММ. СС".

Описание:

Функция ПолучитьВремяТА записывает значение времени *Точки Актуальности* итогов в передаваемые при вызове параметры.

Пример:

```
ПолучитьВремяТА(Час, Минута, Секунда);
```

ПолучитьДокументТА

Получить документ, который стоит на ТА. *Данная функция используется только при наличии компоненты «Оперативный учет».*

Синтаксис:

```
ПолучитьДокументТА()
```

Англоязычный синоним:

```
GetDocOfAP
```

Возвращаемое значение:

Значение типа «документ», который стоит на ТА.

Описание:

Функция ПолучитьДокументТА возвращает документ, который стоит на ТА, если он есть.

Пример:

```
ПоследнийДок = ПолучитьДокументТА();
```

ПолучитьПозициюТА

Получить позицию *Точки Актуальности* итогов. *Данная функция используется только при наличии компоненты «Оперативный учет».*

Синтаксис:

```
ПолучитьПозициюТА()
```

Англоязычный синоним:

```
GetAPPosition
```

Возвращаемое значение:

32-х символьное строковое значение позиции ТА.

Описание:

Функция ПолучитьПозициюТА возвращает 32-х символьное строковое значение позиции ТА.

Пример:

```
ПозицияТА = ПолучитьПозициюТА();
```

УстановитьТАна

Изменить положение *Точки Актуальности* итогов. *Данный метод используется только при наличии компоненты «Оперативный учет». Данный метод используется только в монопольном режиме доступа.*

Синтаксис:

```
УстановитьТАна(<ПоложениеТА>]
```

Англоязычный синоним:

```
SetAPToBeg
```

Параметры:

<ПоложениеТА>	Выражение типа дата, документ или позиция документа, на начало которого устанавливается ТА.

Описание:

Метод УстановитьТАна изменяет положение *Точки Актуальности* итогов на начало даты или документа <ПоложениеТА>. Данный метод используется только в монопольном режиме доступа.

Пример:

```
УстановитьТАна(ВыбДокумент);
```

УстановитьТАпо

Изменить положение *Точки Актуальности* итогов. *Данный метод используется только при наличии компоненты «Оперативный учет». Данный метод используется только в монопольном режиме доступа.*

Синтаксис:
УстановитьТАпо(<ПоложениеТА>)
Англоязычный синоним:
SetAPToEnd
Параметры:

<ПоложениеТА>	Выражение типа дата, документ или позиция документа, на конец которого устанавливается ТА.

Описание:
Метод УстановитьТАпо изменяет положение *Точки Актуальности* итогов на конец даты или документа <ПоложениеТА>. Данный метод используется только в монопольном режиме доступа.
Пример:

```
УстановитьТАпо(ВыбДокумент);
```

Процедуры и функции компоненты «Бухгалтерский учет»

Процедуры и функции этого раздела используются *только при наличии компоненты «Бухгалтерский учет».*

ВыбранныйПланСчетов

Возвращает план счетов, выбранный пользователем в параметрах системы.
Синтаксис:
ВыбранныйПланСчетов()
Англоязычный синоним:
DefaultChartOfAccounts
Возвращаемое значение:
Значение типа «ПланСчетов».
Описание:
Функция ВыбранныйПланСчетов позволяет определить текущее значение параметра «Основной план счетов», выбранное пользователем при работе с системой.
Пример:

```
Сч = СоздатьОбъект("Счет");
Сч.ИспользоватьПланСчетов(ВыбранныйПланСчетов());
```

ОсновнойПланСчетов

Возвращает план счетов, установленный в конфигурации в качестве основного.
Синтаксис:
ОсновнойПланСчетов()
Англоязычный синоним:
MainChartOfAccounts
Возвращаемое значение:
Значение типа «ПланСчетов».
Описание:
Функция ОсновнойПланСчетов выдает план счетов установленный в конфигурации в качестве основного.
Пример:

```
Сч = СоздатьОбъект("Счет");
Сч.ИспользоватьПланСчетов(ОсновнойПланСчетов());
```

СчетПоКоду

Поиск бухгалтерского счета по коду.
Синтаксис:
СчетПоКоду(<КодСчета>[, <ПланСчетов>])
Англоязычный синоним:
AccountByCode
Параметры:

<КодСчета>	Символьное выражение — код счета.
<ПланСчетов>	Значение типа «План Счетов» — план счетов, в котором выполняется поиск. Если параметр не указан, поиск выполняется в основном плане счетов, заданном в метаданных.

Возвращаемое значение:
Значение типа «Счет».
Описание:
Метод СчетаПоКоду позволяет найти значение типа «Счет» по коду счета, указанному в виде символьной строки.
Пример:

```
// При проведении документа формируем проводку
Операция.НоваяПроводка();
Операция.Дебет.Счет = СчетПоКоду("51");
```

НачалоПериодаБИ

Возвращает дату начала периода бухгалтерских итогов.

Синтаксис:

```
НачалоПериодаБИ()
```

Англоязычный синоним:

```
BeginOfPer1odBT()
```

Возвращаемое значение:

Значение типа «Дата» — дата начала периода бухгалтерских итогов.

Описание:

Метод НачалоПериодаБИ возвращает дату начала периода бухгалтерских итогов, заданную в режиме «Настройка параметров системы».

Пример:

```
Дата1 = НачалоПериодаБИ();
```

КонецПериодаБИ

Возвращает дату конца периода бухгалтерских итогов.

Синтаксис:

```
КонецПериодаБИ()
```

Англоязычный синоним:

```
EndOfPer1odBT()
```

Возвращаемое значение:

Значение типа «Дата» — дата конца периода бухгалтерских итогов.

Описание:

Метод КонецПериодаБИ возвращает дату конца периода бухгалтерских итогов, заданную в режиме «Настройка параметров системы».

Пример:

```
Дата2 = КонецПериодаБИ();
```

КонецРассчитанногоПериодаБИ

Возвращает последнюю дату рассчитанных бухгалтерских итогов.

Синтаксис:

```
КонецРассчитанногоПериодаБИ()
```

Англоязычный синоним:

```
EndOfCalculatedPer1odBT()
```

Возвращаемое значение:

Значение типа «Дата» — последняя дата рассчитанных бухгалтерских итогов.

Описание:

Метод КонецРассчитанногоПериодаБИ возвращает последнюю дату рассчитанных бухгалтерских итогов, заданную в режиме «Управление бухгалтерских итогов» при работе в режиме «1С:Предприятие».

Данная функция позволяет определить, до какой даты может быть выполнено обращение к бухгалтерским итогам.

Пример:

```
Если Дата2 > КонецРассчитанногоПериодаБИ() Тогда
   Сообщить("Период не рассчитан!");
   Возврат;
КонецЕсли;
```

НазначитьСчет

Назначает счет значению типа «Вид субконто».

Синтаксис:

```
НазначитьСчет(<ВидСубконто>, <Счет>[, <НомерСубконто>])
```

Англоязычный синоним:

```
SetAccount
```

Параметры:

<ВидСубконто>	Значение типа «ВидСубконто», переданное по ссылке — обычно реквизит диалога формы.
<Счет>	Значение типа «Счет».

<НомерСубконто> Значение типа «Число» — номер субконто для данного счета.

Описание:

Данная процедура используется для организации выбора значения типа «Вид субконто» в диалоге формы применительно к конкретному счету, то есть организовать выбор видов субконто конкретного счета. В качестве первого параметра передается собственно реквизит диалога, а в качестве второго — счет. Вызов данной процедуры ограничивает выбор вида субконто видами, заданными для конкретного счета. Кроме того, само значение вида субконто очищается, если данный вид субконто не используется для указанного счета. Если указан параметр НомерСубконто, то значение вида субконто будет автоматически установлено в соответствии с настройкой субконто указанного счета с данным номером.

Пример:

```
Процедура ПриВыбореСчета()
   НазначитьСчет(ВыбВидСубконто1, Счет, 1);
   НазначитьСчет(ВыбВидСубконто2, Счет, 2);
   НазначитьСчет(ВыбВидСубконто3, Счет, 3);
КонецПроцедуры
```

ВвестиПланСчетов

Выбрать план счетов из существующих планов счетов в диалоге.

Синтаксис:

```
ВвестиПланСчетов(<ПланСчетов>, <Подсказка>, <Таймаут>)
```

Англоязычный синоним:

```
InputChartOfAccounts
```

Параметры:

<ПланСчетов>	Значение типа «План счетов», переданное по ссылке (идентификатор переменной). В данное значение будет осуществлен возврат выбранного значения.
<Подсказка>	Строковое выражение, содержащее строку, которая будет выводиться в заголовке окна диалога.
<Таймаут>	Числовое выражение — интервал времени ожидания ответа пользователя в секундах. Если параметр опущен или равен 0 — ожидание бесконечно. Необязательный параметр.

Возвращаемое значение:

Числовое значение: 1 — выбор осуществлен , 0 — выбор не осуществлен (пользователем нажата кнопка «Отмена», клавиша <Esc> или закрыто окно диалога).

Описание:

Данная функция выводит диалог, в котором пользователь интерактивно выбирает один из существующих планов счетов. Результат выбора возвращается в первый параметр функции.

Пример:

```
Перем ВыбПлан;
Если ВвестиПланСчетов(ВыбПлан, "Введите план счетов") = 1 Тогда
   ...
КонецЕсли;
```

ВвестиВидСубконто

Выбрать вид субконто из существующих видов субконто в диалоге.

Синтаксис:

```
ВвестиВидСубконто(<ВидСубконто>, <Подсказка>, [<Счет>], [<Таймаут>])
```

Англоязычный синоним:

```
InputSubcontoKind
```

Параметры:

<ВидСубконто>	Значение типа «ВидСубконто», переданное по ссылке (идентификатор переменной). В данное значение будет осуществлен возврат выбранного значение вида субконто.
<Подсказка>	Строковое выражение, содержащее строку, которая будет выводиться в заголовке окна диалога выбора вида субконо.
<Счет>	Значение типа «Счет». Если данное значение указано выбор будет ограничен видами субконто, заданными для указанного счета. Необязательный параметр.
<Таймаут>	Числовое выражение — интервал времени ожидания ответа пользователя в секундах. Если параметр опущен или равен 0 — ожидание бесконечно. Необязательный параметр.

Возвращаемое значение:

Числовое значение: 1 — выбор осуществлен , 0 — выбор не осуществлен (пользователем нажата кнопка «Отмена», клавиша <Esc> или закрыто окно диалога).

Описание:

Данная функция выводит диалог, в котором пользователь выбирает один из существующих видов субконто. Результат выбора возвращается в первый параметр функции. Выбор видов субконто может быть ограничен видами субконто, заданными для конкретного счета.

Пример:

```
Перем ВыбВидСубк;
Если ВвестиВидСубконто(ВыбВидСубк, "Введите вид убконто", ВыбСчет) = 1 Тогда
   ...
КонецЕсли;
```

МаксимальноеКоличествоСубконто

Максимальное количество видов субконто, которые могут быть заданы для счетов.

Синтаксис:

```
МаксимальноеКоличествоСубконто()
```

Англоязычный синоним:

```
MaxSubcontoCount
```

Возвращаемое значение:

Число — максимальное количество видов субконто.

Описание:

Метод `МаксимальноеКоличествоСубконто` выдает значение параметра «Максимальное количество субконто», указанное в метаданных в настройке планов счетов. Фактически она позволяет определить, сколько видов субконто может быть задано для счета.

Пример:

```
Если МаксимальноеКоличествоСубконто() < 2 Тогда
   Форма.ВидСубк2.Видимость(0);
КонецЕсли;
```

Процедуры и функции компоненты «Расчет»

ОсновнойЖурналРасчетов

Устанавливает основной журнал расчетов. *Используется только при наличии компоненты «Расчет».*

Синтаксис:

```
ОсновнойЖурналРасчетов(<ЖурналРасч>)
```

Англоязычный синоним:

```
BasicCalcJournal
```

Параметры:

`<ЖурналРасч>`	Строковое значение — идентификатор журнала расчетов, как он задан при конфигурировании.

Возвращаемое значение:

Значение текущего журнала расчетов.

Описание:

Функция `ОсновнойЖурналРасчетов` устанавливает основной журнал расчетов. По умолчанию основной журнал расчетов — первый в списке журналов расчетов.

Пример:

```
ОсновнойЖурналРасчетов("Основной");
```

Глава 8
Системные предопределенные процедуры

Системные предопределенные процедуры предназначены для того, чтобы дать возможность программно обрабатывать системные события, возникающие при интерактивных действиях пользователя, таких как ввод нового, изменение, удаление. Исключение составляют только предопределенные процедуры Модуля документа: `ОбработкаПроведения`, `ОбработкаУдаленияПроведения`, `АрхивироватьДокумент`, которые вызываются как при интерактивном, так и при программном возникновении события.

ВНИМАНИЕ! Системные предопределенные процедуры не являются встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком конфигурации в соответствующих программных модулях.

Вызов системных предопределенных процедур (если они использованы в конфигурации) производится неявно самой системой 1С:Предприятие перед обработкой какого-либо интерактивного действия пользователя, например, при вводе нового документа. В этот момент система передает в предопределенную процедуру фактические значения параметров. В теле процедур переданные параметры могут использоваться для обработки события, например, выдачи предупреждающих сообщений, установки статуса возврата и т. п.

С помощью системной функции `СтатусВозврата` в теле предопределенной процедуры можно устанавливать значение статуса возврата процедуры. Статус возврата используется системой, чтобы определить — выполнить или нет действие того системного события, которое вызвало данную предопределенную процедуру.

Предопределенные процедуры Глобального модуля

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в глобальном программном модуле. Данные процедуры, например, могут использоваться для расширения возможности программного управления правами доступа пользователя к системе.

ПриНачалеРаботыСистемы

Предопределенная процедура при начале работы задачи.

Синтаксис:

`ПриНачалеРаботыСистемы()`

Англоязычный синоним:

`OnStartSystem`

Описание:

Вызов предопределенной процедуры `ПриНачалеРаботыСистемы` производится системой 1С:Предприятие неявно при начале работы с программой. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если время доступа данного пользователя к системе истекло), то пользователь не сможет войти в программу.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриНачалеРаботыСистемы()
   Курс = Константа.ОсновнаяВалюта.Текущ_Курс.Получить(ТекущаяДата();
   Предупреждение("Добро пожаловать! Сегодня " + " " + ТекущаяДата() +
         РазделительСтрок + "Сейчас установлен курс доллара=" + Курс), 7);
КонецПроцедуры
```

См. также: `СтатусВозврата`

ПриЗавершенииРаботыСистемы

Предопределенная процедура при завершении работы задачи.

Синтаксис:

`ПриЗавершенииРаботыСистемы()`

Англоязычный синоним:

`OnFinishSystem`

Описание:

Вызов предопределенной процедуры `ПриЗавершенииРаботыСистемы` производится системой 1С:Предприятие неявно при завершении работы пользователя с системой. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если пользователь не выполнил какие-либо штатные операции), то работа системы не будет завершена, т. е. пользователь не сможет выйти из программы.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриЗавершенииРаботыСистемы()
   Предупреждение("До свидания !", 2);
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУдаленииДокумента

Предопределенная процедура при удалении документа и при отмене пометки на удаление.

Синтаксис:

ПриУдаленииДокумента(<УдалДокум>, <Режим>)

Англоязычный синоним:

OnDeleteDoc

Параметры:

<УдалДокум>	Значение удаляемого документа.
<Режим>	Значение флага непосредственного удаления: 1 — непосредственное удаление; 0 — пометка на удаление.

Описание:

Вызов предопределенной процедуры ПриУдаленииДокумента производится системой 1С:Предприятие неявно при интерактивном удалении документа и в случае отмены пометки на удаление. В этом случае, условия вызова можно узнать методом ПометкаУдаления документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если не истек срок хранения документа), то удаление документа не будет выполнено.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриУдаленииДокумента(Док, Реж)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (Док.Вид() = "РасходнаяНакл") И (Реж = 1) Тогда
         Предупреждение("У вас нет права удалять накладную!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУдаленииЭлемента

Предопределенная процедура при удалении элемента справочника и при отмене пометки на удаление.

Синтаксис:

ПриУдаленииЭлемента(<УдалЭлем>, <Режим>)

Англоязычный синоним:

OnDeleteItem

Параметры:

<УдалЭлем>	Значение удаляемого элемента справочника.
<Режим>	Значение флага непосредственного удаления: 1 — непосредственное удаление; 0 — пометка на удаление.

Описание:

Вызов предопределенной процедуры ПриУдаленииЭлемента производится системой 1С:Предприятие неявно при интерактивном удалении элемента справочника и в случае отмены пометки на удаление. В этом случае, условия вызова можно узнать методом ПометкаУдаления элемента справочника. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если нарушается ссылочная целостность системы), то удаление элемента справочника не будет выполнено.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриУдаленииЭлемента(Элем, Реж)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (Элем.Вид() = "Товар") И (Реж = 1) Тогда
         Предупреждение("У вас нет права удалять Товар!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриОткрытииИстории

Предопределенная процедура при открытии окна «История» значения периодического реквизита элемента справочника или константы.

Синтаксис:

```
ПриОткрытииИстории(<ТипОбъекта>, <Объект>, <ТолькоПросмотр>)
```

Англоязычный синоним:

```
OnOpenHistory
```

Параметры:

<ТипОбъекта>	Название периодического объекта конфигурации, как оно задано в конфигураторе (например, "Справочник.Валюты.ТекущийКурс" или " Константа. ИмяДиректора"). Строковое название агрегатного типа данных может передаваться системой в русском или англоязычном написании (Справочник или Reference, Константа или Const) — в зависимости от установки флага основного языка системы (см. ОсновнойЯзык).
<Объект>	Элемент справочника, для которого выполняется открытие окна истории периодического реквизита. Имеет смысл только для периодических реквизитов справочников, а не для констант.
<ТолькоПросмотр>	Флаг установки режима "только просмотр" для окна истории значения. Если значение этого параметра установить в 1 (в предопределенной процедуре), то окно истории будет открыто только для чтения. Установка значения в 0 — введет режим по умолчанию (определенный правами). Уже открытое окно истории соответствующим образом изменит режим (если это случай, когда процедура обрабатывает не открытие окна, а смену отображаемого объекта). Значение по умолчанию — 0.

Описание:

Вызов предопределенной процедуры ПриОткрытииИстории производится системой 1С:Предприятие неявно при при открытии окна «История» значения периодического реквизита элемента справочника или константы. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю запрещено изменение периодических реквизитов), то запись нового периодического значения не будет выполнена.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриОткрытииИстории(ТипОб, Об, ФлагРежима)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (ТипОб = "Справочник.Валюты.ТекКурс") И
           (Об = Константа.ОсновнаяВалюта)) Тогда
         Предупреждение("У вас нет права просматривать историю курса!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, ОсновнойЯзык

ПриЗаписиИстории

Предопределенная процедура при записи в окне «История» значения периодического реквизита элемента справочника или константы.

Синтаксис:

```
ПриЗаписиИстории(<ТипОбъекта>, <Объект>, <Значение>, <ДатаИстории>)
```

Англоязычный синоним:

```
OnWriteHistory
```

Параметры:

<ТипОбъекта>	Название периодического объекта конфигурации, как оно задано в конфигураторе (например, "Справочник.Валюты.ТекущийКурс" или "Константа. ИмяДиректора"). Строковое название агрегатного типа данных может передаваться системой в русском или англоязычном написании (Справочник или Reference, Константа или Const) — в зависимости от установки флага основного языка системы (см. ОсновнойЯзык).
<Объект>	Элемент справочника, для которого выполняется изменение периодического реквизита. Имеет смысл только для периодических реквизитов справочников, а не для констант.
<Значение>	Значение периодической записи.
<ДатаИстории>	Дата периодической записи.

Описание:

Вызов предопределенной процедуры ПриЗаписиИстории производится системой 1С:Предприятие неявно при интерактивной записи нового периодического значения в окне «История». Если в данной предопределенной процедуре уста-

новить статус возврата — 0 (например, если данному пользователю запрещено изменение периодических реквизитов), то запись нового периодического значения не будет выполнена.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриЗаписиИстории(ТипОб, Об, Значен, ДатаИстор)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (ТипОб = "Справочник.Валюты.ТекКурс") И
           (Об = Константа.ОсновнаяВалюта) И (ДатаИстор > РабочаяДата()) Тогда
         // Записывается курс для валюты, которая записана в Константе
         Предупреждение("У вас нет права устанавливать курс на будущую дату!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

Ом. также: СтатусВозврата, ОсновнойЯзык

ПриУдаленииИстории

Предопределенная процедура при удалении в окне «История» значения периодического реквизита элемента справочника или константы.

Синтаксис:

```
ПриУдаленииИстории(<ТипОбъекта>, <Объект>, <Значение>, <ДатаИстории>)
```

Англоязычный синоним:

```
OnDeleteHistory
```

Параметры:

<ТипОбъекта>	Название периодического объекта конфигурации, как оно задано в конфигураторе (например, "Справочник.Валюты.ТекущийКурс" или " Константа. ИмяДиректора"). Строковое название агрегатного типа данных может передаваться системой в русском или англоязычном написании (Справочник или Reference, Константа или Const) — в зависимости от установки флага основного языка системы (см. ОсновнойЯзык).
<Объект>	Элемент справочника, для которого выполняется изменение периодического реквизита. Имеет смысл только для периодических реквизитов справочников, а не для констант.
<Значение>	Значение периодической записи.
<ДатаИстории>	Дата периодической записи.

Описание:

Вызов предопределенной процедуры ПриУдаленииИстории производится системой 1С:Предприятие неявно при интерактивном удалении периодической записи в окне «История». Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю запрещено изменение периодических реквизитов), то удаления периодического значения не будет выполнено.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриУдаленииИстории(ТипОб, Об, Значен, ДатаИстор)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (ТипОб = "Справочник.Валюты.ТекКурс") И
           (Об=Константа.ОсновнаяВалюта) И (ДатаИстор < РабочаяДата() - 7) Тогда
         // Удаляется курс для валюты, которая записана в Константе
         Предупреждение("У вас нет права удалять прошлый курс!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, ОсновнойЯзык

ПриЗаписиКонстанты

Предопределенная процедура при записи значения константы.

Синтаксис:

```
ПриЗаписиКонстанты(<ИмяКонстанты>, <Значение>)
```

Англоязычный синоним:

```
OnWriteConst
```

Параметры:

<ИмяКонстанты>	Название константы, как оно задано в конфигураторе (например, "СкладПоУмолчанию").
<Значение>	Новое значение константы.

Описание:

Вызов предопределенной процедуры ПриЗаписиКонстанты производится системой 1С:Предприятие неявно при интерактивной записи значения константы. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данное значение константы запрещено), то запись константы не будет выполнена.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриЗаписиКонстанты(ИмяКонстанты, Значение)
   Если ИмяКонстанты = "ТекущийПрефикс" Тогда
      Если НазваниеНабораПрав() = "Продавец" Тогда
         Предупреждение("У вас нет права менять префикс!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриОтмененПроведенияДокумента

Предопределенная процедура при отмене проведения документа.

Синтаксис:

ПриОтмененПроведенияДокумента(<Докум>)

Англоязычный синоним:

OnUnPosting

Параметры:

<Докум> Значение обрабатываемого документа.

Описание:

Вызов предопределенной процедуры ПриОтмененПроведенияДокумента производится системой 1С:Предприятие неявно при интерактивной попытке выполнить операцию «Сделать непроведенным». Если в данной предопределенной процедуре установить статус возврата — 0 (например, если дата документа меньше некоторой константы), то отмена проведения документа не будет выполнена.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриОтмененПроведенияДокумента(Док)
   Если Док.ДатаДок <= Константа.ДатаЗащитыДокументов Тогда
      Предупреждение("Нельзя снимать с проведения архив!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, СделатьНеПроведенным

ПриИзмененииВремениДокумента

Предопределенная процедура при изменении времени документа.

Синтаксис:

ПриИзмененииВремениДокумента(<Докум>)

Англоязычный синоним:

OnChangeTimeDoc

Параметры:

<Докум> Значение обрабатываемого документа.

Описание:

Вызов предопределенной процедуры ПриИзмененииВремениДокумента производится самой системой 1С:Предприятие неявно при интерактивной попытке изменить время документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если время данного документа нельзя менять), изменение времени документа не будет выполнено.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриИзмененииВремениДокумента(Док)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права менять время документа!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУстановкеОтбора

Предопределенная процедура при установке отбора.

Синтаксис:

```
ПриУстановкеОтбора(<ИмяРеквизОтбора>, <Значение>)
```

Англоязычный синоним:

```
OnSetSelectInJournal
```

Параметры:

<ИмяРеквизОтбора>	Строковое значение — название общего реквизита документа (как оно задано в конфигураторе), по которому производится отбор (например, "Автор").
<Значение>	Значение реквизита отбора. Допустим, у документов существует общий реквизит "Автор", по которому решено провести отбор, значит в этом параметре будет передано конкретное значение этого реквизита, по которому решено провести отбор (например, "Сидоров И.А.").

Описание:

Вызов предопределенной процедуры ПриУстановкеОтбора производится самой системой 1С:Предприятие неявно при интерактивной попытке установить отбор документов в журнале. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя устанавливать данное значение отбора документов), установка не будет выполнена.

Данная предопределенная процедура может располагаться в глобальном программном модуле и модуле формы журнала. Если данная процедура присутствует в модуле формы соответствующего журнала, то будет вызвана она, если нет, то будет вызвана процедура из глобального модуля.

Пример:

```
Процедура ПриУстановкеОтбора(ИмяОтбора, ЗначОтбора)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если (Имя0тбора = "Автор") И (ЗначОтбора <> ТекущПользователь) Тогда
         Предупреждение("У вас нет права просматривать чужие документы!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриСменеРасчетногоПериода

Предопределенная процедура, вызываемая при смене текущего расчетного периода журнала расчетов.

Синтаксис:

```
ПриСменеРасчетногоПериода(<ЖурналРасчетов>, <Период>)
```

Англоязычный синоним:

```
OnPer1odChange
```

Параметры:

<ЖурналРасчетов>	Журнал расчетов, период которого изменяется (агрегатный объект типа «ЖурналРасчетов»).
<Период>	Устанавливаемый расчетный период (значение типа «ПериодРасчета»).

Описание:

Вызов предопределенной процедуры ПриСменеРасчетногоПериода на исполнение производится системой 1С:Предприятие неявно при попытке смены текущего расчетного периода любого журнала расчетов. Если в данной предопределенной процедуре установить статус возврата — 0, то текущий расчетный период не будет изменен.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриСменеРасчетногоПериода(ЖР, Период)
   Если ЖР.Вид() = "Зарплата" Тогда
      Предупреждение("Это смена текущего расчетного периода для журнала
                        | Зарплата");
   КонецЕсли;
   Если Период.ДатаНачала <= ЖР.НачалоТекущегоПериода Тогда
      //не позволяем откатывать период назад
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУдаленииСчета

Предопределенная процедура, выполняемая при удалении бухгалтерского счета.

Синтаксис:

```
ПриУдаленииСчета(<УдалСчет>, <Режим>)
```

Англоязычный синоним:

```
OnDeleteAccount
```

Параметры:

<УдалСчет>	Значение типа «Счет» — удаляемый бухгалтерский счет.
<Режим>	Режим удаления. Может принимать значения: 1 — счет будет удален; 0 — счет будет помечен на удаление.

Описание:

Вызов предопределенной процедуры ПриУдаленииСчета производится системой 1С:Предприятие неявно при интерактивном удалении счета из плана счетов. В качестве параметров в процедуру передаются значение удаляемого счета и режим удаления: непосредственное удаление или пометка на удаление.

Если в данной процедуре установить статус возврата 0 (например, если нарушается ссылочная целостность системы), то счет не будет удален (или помечен на удаление).

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриУдаленииСчета(УдСчет)
   Если УдСчет.Валютный() = 1 Тогда
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыклВклПроводокОперации

Предопределенная процедура, выполняемая при выключении или включении проводок.

Синтаксис:

```
ПриВыклВклПроводокОперации(<Документ>)
```

Англоязычный синоним:

```
OnOperEntryOffOn
```

Параметры:

<Документ>	Значение типа «Документ» — документ, которому принадлежит операция.

Описание:

Вызов предопределенной процедуры ПриВыклВклПроводокОперации производится системой 1С:Предприятие неявно при интерактивном выключении или включении проводок. В качестве параметра в процедуру передается значение документа, которому принадлежит операция.

Если в данной процедуре установить статус возврата 0, то действие не будет выполнено.

Данная предопределенная процедура может располагаться только в глобальном программном модуле.

Пример:

```
Процедура ПриВыклЗклПроводокОперации(Док)
   Если Док.Вид() <> "Операция" Тогда
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 9
Работа с Константами

Константа — это агрегатный тип данных, средство работы с постоянными (условно постоянными) величинами. В константах хранится информация, характеризующая конфигурацию в целом, которая может быть как постоянной, так и изменяться с той или иной периодичностью.

Константы могут быть периодическими или нет (это задается в конфигураторе при их создании). Периодические константы — это константы, значения которых связаны с датой. При изменении значения периодической константы старое значение сохраняется, просто новое значение действует с указанной даты, а старое до указанной даты.

Пример:

* Изначально значение периодической константы Константа.ИмяДиректора было — "Иванов И. И.". В некоторый момент директор сменился.

```
Константа.ИмяДиректора.Установить('01.05.96', "Петров П.П.");
```

* Данный оператор устанавливает новое значение периодической константы с даты '01.05.96', но старое значение — "Иванов И. И." сохраняется до этой даты и может быть получено, например, следующей строкой текста программы:

```
ПрежнДиректор = Константа.ИмяДиректора.Получить('01.01.96');
```

Средства языка предоставляют возможность непосредственного доступа к созданным в конфигураторе названиям констант в любом программном модуле, т. к. константы принадлежат глобальному контексту задачи. Работа с периодическими константами осуществляется при помощи методов Получить и Установить или при помощи специального агрегатного типа данных — *Периодический*. Работа с непериодическими константами осуществляется непосредственно через их идентификаторы.

Использование непериодических констант в синтаксисе языка аналогично глобальным переменным, т. е. идентификаторы непериодических констант могут размещаться в левой и правой части оператора присваивания, в выражениях, быть параметрами методов, процедур или функций в любом программном модуле.

В качестве имени константы обязательно должно выступать полное имя конкретной константы, как оно объявлено в конфигураторе. Имя константы записывается через точку после ключевого слова Константа, т. е. полное имя константы записывается следующим образом:

```
Константа.<Имя_константы>,
```

где <Имя_константы> — название конкретной константы, как оно объявлено в окне «Метеданные» конфигуратора.

Англоязычный синоним ключевого слова Константа — Const.

Пример:

* Пример работы с непериодическими константами:

```
Константа.Организация = "АО СПЕЦСТРОЙКОНСЕРВБАНК";
Константа.Адрес = "г.Москва, Вернадского, 523, 25";
Р_Счет = Константа.НашСчет;
// операторы
Если Всего > Константа.МинимальнаяЗарплата Тогда
   // операторы
Иначе
   // операторы
КонецЕсли;
```

Методы констант

НазначитьТип

Назначить тип для константы неопределенного типа.

Синтаксис:

```
НазначитьТип(<ИмяКонстанты>, <ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
SetType
```

Параметры:

<ИмяКонстанты>	Строковое выражение — название константы неопределенного типа, как она названа в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается константе. Например: "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных.

	Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:

Метод НазначитьТип позволяет назначить тип для константы, которой в конфигураторе назначен тип «Неопределенный».

Пример:

```
Константа.НазначитьТип("ТМЦ", "Справочник.Товары");
```

УстановитьАтрибут

Установить значение константы по имени идентификатора.

Синтаксис:

```
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SetAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя константы, как оно задано в конфигураторе.
<Значение>	Выражение, содержащее устанавливаемое значение константы.

Описание:

Метод УстановитьАтрибут позволяет установить значение константы по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
Константа.УстановитьАтрибут("ФлагЗапретаРедактирования", 1);
```

ПолучитьАтрибут

Получить значение константы по идентификатору.

Синтаксис:

```
ПолучитьАтрибут(<ИмяАтрибута>)
```

Англоязычный синоним:

```
GetAttrib
```

Параметры:

<ИмяАтрибута>	Строковое выражение, содержащее имя константы, как оно задано в конфигураторе.

Возвращаемое значение:

Значение константы <ИмяАтрибута>.

Описание:

Метод ПолучитьАтрибут позволяет получить значение константы по идентификатору, как оно задано в метаданных.

Пример:

```
ФлЗапрРед = Константа.ПолучитьАтрибут("ФлагЗапретаРедактирования");
```

Методы периодических констант

Получить

Получить значение периодической константы на дату.

Синтаксис:

```
Получить(<Дата>)
```

Англоязычный синоним:

```
GetValue
```

Параметры:

<Дата>	Выражение со значением требуемой даты.

Возвращаемое значение:

Значение константы на заданную дату.

Описание:

Метод Получить возвращает значение константы на определенную дату. Данный метод можно использовать только для периодических констант.

Пример:

```
// Константа.РазмерКомпенсации – периодическая
// поэтому получим значение этой константы на дату документа
РазмерДК = Константа.РазмерКомпенсации.Получить(ДатаДок);
Если РазмерДК <> 0 Тогда
   Результат = РазмерДК;
```

```
Иначе
   // операторы
КонецЕсли;
```

Установить

Установить значение периодической константы на дату.

Синтаксис:

```
Установить(<Дата>, <Значение>)
```

Англоязычный синоним:

```
SetValue
```

Параметры:

<Дата>	Выражение со значением требуемой даты.
<Значение>	Новое значение константы.

Описание:

Метод `Установить` устанавливает значение константы на определенную дату. Данный метод можно использовать только для периодических констант.

Пример:

```
Константа.РасчетныйСчет.Установить('01.07.96', "77889001");
Константа.Директор.Установить('01.01.85', "Иванов П.С.")
```

Глава 10
Работа со Справочниками

Справочники — это агрегатный тип данных, средство для работы со списками однородных элементов данных. При помощи справочников организуется ввод стандартной информации в документы, а также ее просмотр и, если необходимо, корректировка. В большинстве своем справочники являются электронными аналогами каталогов. Каждая карточка — это строка справочника, а сведения, заносимые в карточку, являются реквизитами справочника.

Например, для того, чтобы покупатель, продавец, кладовщик, директор однозначно понимали, о каком товаре идет речь, каждый должен называть его одинаково, т. е. в соответствии с однажды утвержденным справочником товаров. Обычно в торговом предприятии он имеет вид прайс-листа, а если такой справочник товаров хранится в компьютере, то в него заносят всю возможную номенклатуру товаров, с которыми работает торговая фирма.

Название и структура каждого конкретного справочника определяется при его создании в конфигураторе. У любого справочника существует два обязательных реквизита, которые создаются автоматически — "Код" и "Наименование". Другие реквизиты справочника, которые могут содержать различную дополнительную информацию помимо наименования, определяются в конфигураторе конкретно для каждого создаваемого справочника.

Реквизиты справочников могут быть периодическими или нет (это задается в конфигураторе при создании реквизита). Периодические реквизиты — это реквизиты, значения которых связаны с датой. При изменении значения периодического реквизита старое значение сохраняется, при этом новое значение действует с указанной даты, а старое — до указанной даты. Работа с периодическими реквизитами осуществляется при помощи методов Получить и ИспользоватьДату, а также при помощи специального агрегатного типа данных — *Периодический*.

Система 1С:Предприятие дает возможность создавать и использовать многоуровневые справочники. Количество уровней вложенности конкретного справочника устанавливается в конфигураторе. Кроме того, справочникам можно указывать их подчиненность другим справочникам, образовывая таким образом связи типа «один ко многим».

Контекст работы со справочниками

В синтаксисе языка обращение к атрибутам, а также вызов методов справочников зависит от контекста выполнения программного модуля.

Если конкретный элемент или группа справочника входит (согласно локального контекста) в набор непосредственно доступных модулю значений агрегатных типов данных (см. «Виды программных модулей»), то обращение к атрибуту, вызов метода для этого элемента или группы справочника — просто имя этого атрибута или метода с указанием необходимых параметров.

Пример:

* В форме редактирования элемента справочника «Сотрудники» мы имеем непосредственный доступ к текущему элементу (сотруднику) справочника. Значит, чтобы изменить имя текущего сотрудника, просто укажем:

```
Наименование = "Сидоров С.С.";
```

Значение элемента или группы справочника может быть получено из других источников, например как реквизит какого-либо документа. Чтобы получить доступ к атрибуту, вызвать метод такой переменной со значением типа «Справочник», имя этого атрибута, метода (с указанием необходимых параметров) пишется через точку после имени реквизита.

Пример:

* Если в документе значение реквизита "Сотрудники" имеет тип «Справочник», имя сотрудника можно узнать следующим образом:

```
ИмяСотр = Док.Сотрудники.Наименование;
```

В других случаях доступ к атрибутам, вызов методов конкретного элемента или группы справочника происходит при помощи переменной со ссылкой на объект типа «Справочник». Объект создается функцией СоздатьОбъект, ссылка на который присваивается переменной. Чтобы вызвать атрибут или метод объекта, имя этого атрибута, метода (с указанием необходимых параметров) пишется через точку после имени переменной.

При создании переменной со ссылкой на объект типа «Справочник» в качестве имени справочника должно выступать полное название конкретного вида справочника, как он объявлен в конфигураторе. Вид справочника записывается через точку после ключевого слова " Справочник", т. е. полное имя справочника записывается следующим образом:

```
Справочник.Имя_Справочника
```

где <Имя_Справочника> — название справочника, определенное в конфигураторе.

Англоязычный синоним ключевого слова Справочник — Reference. Допускается создавать объект неопределенного вида справочника. В этом случае название конкретного вида справочника в параметре вызова функции СоздатьОбъект опускается. Далее для работы с таким объектом ему надо установить вид справочника при помощи метода Вид.

Пример:

```
Спр1=СоздатьОбъект("Справочник.Товары");
Спр2=СоздатьОбъект("Справочник.Клиенты");
Спр3=СоздатьОбъект("Справочник");
Спр3.Вид("Валюты");
```

Замечание. Следует обратить особое внимание, что переменной может быть присвоена ссылка на позиционируемый объект или само значение элемента справочника (например, переменной может быть присвоено значение реквизита документа, который имеет тип «Справочник»). Использование ссылки, созданной при помощи функции СоздатьОбъект существенно отличается от работы непосредственно со значением элемента справочника. Только при работе со ссылкой на позиционируемый объект типа «Справочник» есть возможность изменять позиционирование (найти-выбрать...) текущего элемента справочника (т. е. осуществлять навигацию по справочнику), создавать новые элементы и удалять существующие. С другой стороны, ссылка не содержит собственно значения элемента справочника, которое можно присвоить чему-либо. Однако, его всегда можно получить используя метод `ТекущийЭлемент`.

Замечание. Объект, созданный при помощи функции `СоздатьОбъект`, изначально не спозиционирован, т. е. не указывает на конкретный элемент справочника. Чтобы начать с ним работать, его предварительно надо позиционировать при помощи методов `НайтиЭлемент`, `НайтиПоКоду`, `ПолучитьЭлемент` и т. п.

Пример:

```
*
//В модуле Формы списков Справочника
// меняем наименование выбранного в списке элемента справочника
Наименование = "Администрация";
*
// В модуле Формы элемента справочника или
// В модуле Формы группы справочника
// при редактирования одного элемента справочника
// меняем наименование обрабатываемого элемента справочника
Наименование = "Бухгалтерия";
*
// В других программных модулях
Спр = СоздатьОбъект("Справочник.Товары");
Спр.Новый();
//задаем реквизиты элемента Справочника, используя атрибуты
Спр.Наименование = "Авто-Элемент";
Спр.Код = 1032;
Спр.Размер = 10045;
Спр.Записать();
```

Атрибуты справочников

Код

Значение реквизита Код.
Синтаксис:
`Код`
Англоязычный синоним:
`Code`
Описание:
Атрибут `Код` предоставляет доступ к значению обязательного реквизита " Код" выбранного элемента справочника.
Пример:

```
Спр = СоздатьОбъект("Справочник.Подразделения");
Спр.НайтиПоКоду("29");
Ид = Спр.Код;
Спр.Код = Ид + "5";
```

Наименование

Значение реквизита `Наименование`.
Синтаксис:
`Наименование`
Англоязычный синоним:
`Description`
Описание:

Атрибут Наименование предоставляет доступ к значению обязательного реквизита "Наименование" выбранного элемента справочника.

Пример:

```
Спр = СоздатьОбъект("Справочник.Подразделения");
Спр.НайтиПоКоду("29");
Подразделение = Спр.Наименование;
```

<Реквизит>

Значение реквизита справочника.

Синтаксис:

<Реквизит> Идентификатор реквизита справочника, как он задан в конфигураторе.

Описание:

Атрибут <Реквизит> предоставляет доступ к значению реквизита выбранного элемента справочника. В тексте программного модуля в качестве названия реквизита подставляется идентификатор конкретного реквизита справочника, созданного в конфигураторе.

Пример:

```
// В этом примере справочник "Сотрудники" имеет реквизит "Оклад"
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.НайтиПоКоду("111");
ОкладСотр = Спр.Оклад;
Спр.Оклад = ОкладСотр * 1.5;
```

Родитель

Значение родительской группы элемента справочника.

Синтаксис:

```
Родитель
```

Англоязычный синоним:

```
Parent
```

Описание:

Атрибут Родитель предоставляет доступ к значению родительской группы выбранного элемента справочника.

Данный атрибут может быть изменен только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Функция ДатьРодителя(Элем)
   // Справочник Товаров многоуровневый
   // Получая в параметре функции значение товара,
   // надо выдать имя группы товара
   Спр = СоздатьОбъект("Справочник.Товары");
   Спр.НайтиЭлемент(Элем);
   Если Спр.Уровень() > 1 Тогда
      Спр.НайтиЭлемент(Спр.Родитель);
      Возврат "Этот товар принадлежит группе " + Спр.Наименование;
   Иначе
      Возврат "Это товар первого уровня - нет родителя!";
   КонецЕсли;
КонецФункции
```

Владелец

Значение элемента сопряженного справочника, которому подчинен выбранный элемент текущего справочника.

Синтаксис:

```
Владелец
```

Англоязычный синоним:

```
Owner
```

Описание:

Атрибут Владелец предоставляет доступ к значению элемента сопряженного справочника, которому подчинен выбранный элемент текущего справочника.

Данный атрибут может быть изменен только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Функция ДатьВладельца(Элем)
   // Справочник "Единицы" подчинен Справочнику "Товары"
   // Получая в параметре функции значение Единицы измерения товара,
   // надо выдать имя товара, для которого задана эта единица изм.
```

```
    СпрТ = СоздатьОбъект("Справочник.Товары");
    СпрЕ = СоздатьОбъект("Справочник.Единицы");
    СпрЕ.НайтиЭлемент(Элем);
    СпрТ.НайтиЭлемент(СпрЕ.Владелец);
    Возврат "Это единица измерения товара " + СпрТ.Наименование;
КонецФункции
```

Методы периодических реквизитов

Получить

Получить значение периодического реквизита справочника на дату (или на документ).

Синтаксис:

`Получить(<Дата>)`

Англоязычный синоним:

`GetValue`

Параметры:

`<Дата>`	Необязательный параметр. Выражение со значением требуемой даты или значение типа «документ» (в этом случае возвращается значение на дату и время документа). Значение по умолчанию — ТА.

Возвращаемое значение:

Значение периодического реквизита справочника на заданную дату (или на документ).

Описание:

Метод `Получить` возвращает значение периодического реквизита справочника на определенную дату или на документ. Данный метод разрешено использовать только для периодических реквизитов справочников, если для справочника не установлена дата при помощи метода `ИспользоватьДату`.

Замечание: Если к объекту типа «справочник» однажды применен метод `ИспользоватьДату`, то в дальнейшем, чтобы выбирать значения периодических реквизитов справочника, нельзя применять к этому же объекту метод `Получить`, т. е. в такой последовательности эти два метода несовместимы.

Пример:

```
// при работе в контексте документа, его реквизит "Сотрудник"
// является переменной типа "справочник",
// чтобы получить значение периодического реквизита "Оклад" этого
// справочника, применим функцию Получить
ОкладСотр = Сотрудники.Оклад.Получить(ДатаДок);
```

См. также: `ИспользоватьДату`, `Установить`

Установить

Записать новое значение периодического реквизита справочника на заданную дату.

Синтаксис:

`Установить(<Дата>, <Значение>)`

Англоязычный синоним:

`SetValue`

Параметры:

`<Дата>`	Выражение со значением требуемой даты.
`<Значение>`	Выражение, содержащее устанавливаемое значение периодического реквизита справочника.

Описание:

Метод `Установить` записывает значение периодического реквизита справочника на определенную дату. Данный метод разрешено использовать только для периодических реквизитов справочников, если для справочника не установлена дата при помощи метода `ИспользоватьДату`.

Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.

Замечание: Если к объекту типа «справочник» однажды применен метод `ИспользоватьДату`, то в дальнейшем, чтобы записать новые значения периодических реквизитов справочника, нельзя применять к этому же объекту метод `Установить`, т. е. в такой последовательности эти два метода несовместимы.

Пример:

```
// при работе в контексте документа, его реквизит "Сотрудник"
```

```
// является переменной типа "справочник",
// чтобы установить новое значение периодического
// реквизита "Оклад" этого элемента
// справочника, применим функцию Получить
Сотрудники.Оклад.Установить(ДатаДок, МаксОклад);
```

См. также: ИспользоватьДату, УстановитьРеквизитСправочника, Получить

Методы справочников

Вид

Определить вид справочника.

Синтаксис:

```
Вид(<Название>)
```

Англоязычный синоним:

```
Kind
```

Параметры:

<Название>	Необязательный параметр. Строковое выражение с названием вида справочника.

Возвращаемое значение:

Строковое значение, содержащее текущее название вида справочника (на момент до исполнения метода).

Описание:

Метод Вид позволяет установить или получить текущее название вида справочника. В тексте программы метод Вид можно использовать как процедуру или как функцию. Если при вызове метода параметр <Название> задан явно, то вид справочника устанавливается в соответствии с этим параметром. Метод возвращает строку, содержащую текущий (на момент до исполнения метода) идентификатор вида справочника, как он задан в конфигураторе.

Устанавливать новое значение вида справочника допускается только для объектов типа «Справочник» неопределенного вида, созданных при помощи функции СоздатьОбъект.

Пример:

```
// отобразим вид справочника в строке состояния
Спр1 = СоздатьОбъект("Справочник.Товары");
Состояние(Спр1.Вид());
Спр2 = СоздатьОбъект("Справочник.Клиенты");
Состояние(Спр2.Вид());
Спр3 = СоздатьОбъект("Справочник");
Спр3.Вид("Валюты");
Состояние(Спр3.Вид());
```

ПредставлениеВида

Определить пользовательское представление вида справочника.

Синтаксис:

```
ПредставлениеВида()
```

Англоязычный синоним:

```
KindPresent
```

Возвращаемое значение:

Строковое значение, содержащее пользовательское представление вида справочника (синоним справочника или, если он пустой, то идентификатор).

Описание:

Метод ПредставлениеВида позволяет получить пользовательское представление вида справочника, как он задан в конфигураторе.

Пример:

```
// отобразим представление вида справочника в строке состояния
Спр1 = СоздатьОбъект("Справочник.Товары");
Состояние(Спр1.ПредставлениеВида());
```

Уровень

Определить номер уровня элемента справочника.

Синтаксис:

```
Уровень()
```

Англоязычный синоним:

```
Level
```

Возвращаемое значение:

Числовое значение номера уровня текущего элемента справочника.

Описание:

Метод Уровень возвращает номер уровня текущего элемента справочника в структуре дерева многоуровневого справочника.

Пример:

```
Спр = СоздатьОбъект("Справочник.Подразделения");
Спр.ИспользоватьДату(РабочаяДата());
Спр.ВыбратьЭлементы();
Пока Спр.ПолучитьЭлемент() > 0 Цикл
   Сообщить(Строка(Спр.Наименование + Спр.Код + Спр.Уровень());
КонецЦикла;
```

УстановитьАтрибут

Установить значение реквизита по имени идентификатора.

Синтаксис:

```
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SetAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.
<Значение>	Выражение, содержащее устанавливаемое значение реквизита.

Описание:

Метод УстановитьАтрибут позволяет установить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
Тов.УстановитьАтрибут("ЦенаРозн", ЦенаТов);
```

ПолучитьАтрибут

Получить значение реквизита по имени идентификатора.

Синтаксис:

```
ПолучитьАтрибут(<ИмяРеквизита>)
```

Англоязычный синоним:

```
GetAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.

Возвращаемое значение:

Значение реквизита <ИмяРеквизита>.

Описание:

Метод ПолучитьАтрибут позволяет получить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
ЦенаТов = Тов.ПолучитьАтрибут("ЦенаРозн");
```

ЭтоГруппа

Проверить принадлежность к группам.

Синтаксис:

```
ЭтоГруппа()
```

Англоязычный синоним:

```
IsGroup
```

Возвращаемое значение:

Числовое значение: 1 — если выбранный элемент справочника является группой , 0 — если выбранный элемент справочника обычный элемент.

Описание:

Метод ЭтоГруппа возвращает числовое значение 1 — если выбранный элемент справочника является группой , 0 — если выбранный элемент справочника обычный элемент.

Пример:

* В данном примере мы просматриваем весь справочник "Сотрудники" и для групп сотрудников выводим в таблицу секцию "Группа", а для сотрудников выводим секцию " Элемент"

```
Таб = СоздатьОбъект("Таблица");
Таб.ВывестиСекцию("Отчет");
```

```
Сотр = СоздатьОбъект("Справочник.Сотрудники");
Сотр.ВыбратьЭлементы();
Пока Сотр.ПолучитьЭлемент() > 0 Цикл
   Если Сотр.ЭтоГруппа() = 1 Тогда
      Таб.ВывестиСекцию("Группа");
   Иначе
      Таб.ВывестиСекцию("Элемент");
   КонецЕсли;
КонецЦикла;
```

См. также: ПринадлежитГруппе

ПринадлежитГруппе

Проверить принадлежность к группе.

Синтаксис:

ПринадлежитГруппе(<Группа>)

Англоязычный синоним:

BelongsToGroup

Параметры:

<Группа>	Выражение со значением группы справочника.

Возвращаемое значение:

Числовое значение: 1 — если выбранный элемент справочника принадлежит указанной группе, 0 — если нет.

Описание:

Метод ПринадлежитГруппе проверяет, принадлежит ли указанной группе текущий элемент справочника (не важно, на каком нижестоящем уровне он находится). Возвращаемое числовое значение: 1 — если да; 0 — если нет.

Пример:

```
// выведем в отчет только те подразделения, которые
// принадлежат группе ВыбГруппа
Процедура ВывестиНаПечать()
   // . . .
   Возврат;
КонецПроцедуры
...
Спр = СоздатьОбъект("Справочник.Подразделения");
// Задаем выборку в порядке кодов Спр.ПорядокКодов();
Спр.ВыбратьЭлемернты();
Пока Спр.ПолучитьЭлемент() = 1 Цикл
   Если Спр.ПринадлежитГруппе(ВыбГруппа) = -1 Тогда
      ВывестиНаПечать();
   КонецЕсли;
КонецЦикла;
```

См. также: ЭтоГруппа

Выбран

Проверить факт выбора элемента справочника.

Синтаксис:

Выбран()

Англоязычный синоним:

Selected

Возвращаемое значение:

Числовое значение: 1 — если элемент справочника выбран (спозиционирован); 0 — если не выбран.

Описание:

Метод Выбран возвращает число со значением 1 — если элемент справочника выбран (спозиционирован), 0 — если элемент справочника не выбран.

Пример:

```
// В диалоге формирования некоторого отчета
// ВыбСклад - реквизит диалога типа «справочник.Склад»
Если ВыбСклад.Выбран() = 0 Тогда
   // Если Склад в диалоге не выбран, то формируем без условий
   Заг1 = "По всем складам.";
Иначе
   // если в диалоге выбран Склад
```

```
   // то формируем отчет только по выбранному складу
   Заг1 = "Отчет по складу " + ВыбСклад.Наименование;
КонецЕсли;
```

Выбрать

Выбрать элемент справочника в диалоге.

Синтаксис:

```
Выбрать(<Подсказка>, <Форма Списка>)
```

Англоязычный синоним:

```
Choose
```

Параметры:

<Подсказка>	Текст заголовка окна диалога выбора. Может использоваться в качестве подсказки конечному пользователю.
<ФормаСписка>	Строковое выражение идентификатора той формы списка справочника (как он объявлен в конфигураторе), которая должна использоваться для выбора. Если это значение пустое, то используется форма списка по умолчанию.

Возвращаемое значение:

Числовое значение: 1 — если элемент выбран; 0 — если не выбран.

Описание:

Метод Выбрать вызывает диалоговое окно для выбора элемента и затем позиционирует справочник на выбранном элементе. Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
// данная процедура присваивает некоторому реквизиту "Фирма"
// значение из Справочника.Фирмы
Процедура УстФирмы()
   // Создадим объект требуемого справочника
   Фрм = СоздатьОбъект("Справочник.Фирмы");
   // реквизиты справочника могут быть периодическими
   // поэтому установим дату выборки периодических реквизитов
   Фрм.ИспользоватьДату(ДатаДок);
   // Вызываем диалог выбора элемента Справочника
   Если Фрм.Выбрать("Выберите фирму", "") > 0 Тогда
      Фирма = Фрм.ТекущийЭлемент();
   КонецЕсли;
КонецПроцедуры
```

См. также: ВидыДляВыбора

ВидыДляВыбора

Установка выбираемых видов для справочника неопределенного вида.

Синтаксис:

```
ВидыДляВыбора(<СписокВидов>)
```

Англоязычный синоним:

```
KindsForChoise
```

Параметры:

<СписокВидов>	Необязательный параметр. Строковое выражение, содержащее список видов выбираемых справочников, разделенных запятыми.

Возвращаемое значение:

Строковое значение, содержащее текущий список видов выбираемых справочников, разделенных запятыми (на момент до исполнения метода).

Описание:

Метод ВидыДляВыбора устанавливает выбираемые виды для объекта-справочника неопределенного вида. Данный метод обычно используется до начала интерактивного позиционирования элемента справочника, например, при помощи метода Выбрать.

Данный метод может использоваться только для объектов-справочников неопределенного вида, либо созданных функцией СоздатьОбъект, либо определенных в конфигураторе как реквизиты диалога или другого объекта. Если метод ВидыДляВыбора применен к реквизиту диалога типа «справочник неопределенного вида», то интерактивный выбор элемента справочника будет производиться только среди установленных видов справочников.

Пример:

```
// данная процедура присваивает некоторому реквизиту "Фирма"
// значение из Справочника.Фирмы
Процедура УстФирмы()
```

```
   // Создадим объект требуемого справочника
   Фрм = СоздатьОбъект("Справочник");
   // реквизиты справочника могут быть периодическими
   // поэтому установим дату выборки периодических реквизитов
   Фрм.ВидыДляВыбора("Фирмы");
   Фрм.ИспользоватьДату(ДатаДок);
   // Вызываем диалог выбора элемента справочника
   Если Фрм.Выбрать("Выберите фирму", "") > 0 Тогда
      Фирма = Фрм.ТекущийЭлемент();
   КонецЕсли;
КонецПроцедуры
```

См. также: Выбрать

ВыборГруппы

Установить режим выборки групп.

Синтаксис:

ВыборГруппы(<Режим>)

Англоязычный синоним:

SelectGroup

Параметры:

<Режим> Необязательный параметр. Числовое выражение:
1 — выбирать группы; 0—не выбирать группы.

Возвращаемое значение:

Текущее числовое значение режима выборки групп (на момент до исполнения метода).

Описание:

Метод ВыборГруппы устанавливает режим выборки групп. Данный метод может применяться как для позиционируемых объектов, созданных функцией СоздатьОбъект, так и для элементов диалога для полей типа «справочник» (см. «Методы элементов диалога»).

По умолчанию, выборка элементов справочников для полей в формах документов, журналов и справочников установлена без выбора групп, а в форме отчета с выбором групп. Поэтому реально имеет смысл применять данный метод только в том случае, если надо изменить режим выборки групп.

Пример:

* Если в форме документа необходимо, чтобы какой-либо реквизит "Статус" типа «справочник» мог принимать значения как элемента, так и группы, то этому реквизиту следует установить:

```
Статус.ВыборГруппы(1);
*
Спр = СоздатьОбъект("Справочник.Сотрудники");
// Задаем выборку без групп
Спр.ВыборГруппы(0);
// Открываем выборку
Спр.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока (Спр.ПолучитьЭлемент() > 0) Цикл
   Состояние(Спр.Наименование);
КонецЦикла;
```

ТекущийЭлемент

Получить значение элемента справочника.

Синтаксис:

ТекущийЭлемент()

Англоязычный синоним:

Currentltem

Возвращаемое значение:

Значение элемента справочника.

Описание:

Метод ТекущийЭлемент возвращает значение элемента справочника в целом, как объекта. Данный метод применяется, например, если нужно элемент справочника передать как параметр в вызове какого-либо метода или присвоить какому-либо реквизиту.

Пример:

```
// данная процедура присваивает некоторому реквизиту "Фирма"
// значение из справочника "Фирмы"
```

```
Процедура УстФирмы()
   // Создадим объект требуемого справочника
   Фрм = СоздатьОбъект("Справочник.Фирмы");
   // Вызываем диалог выбора элемента справочника
   Если Фрм.Выбрать("Выберите фирму", "") > 0 Тогда
      Фирма = Фрм.ТекущийЭлемент();
   КонецЕсли;
КонецПроцедуры
```

ПолныйКод

Определить полный код элемента справочника.

Синтаксис:

`ПолныйКод()`

Англоязычный синоним:

`FullCode`

Возвращаемое значение:

Строковое значение, содержащее полный код выбранного элемента справочника.

Описание:

Метод `ПолныйКод` возвращает строку, содержащую полный код выбранного элемента справочника (с кодами групп всех вышестоящих уровней, разделенными символом "/").

Пример:

```
ПолнКод = Спр.Подразделения.ПолныйКод();
```

См. также: `НайтиПоКоду`

ПолноеНаименование

Определить полное наименование элемента справочника.

Синтаксис:

`ПолноеНаименование()`

Англоязычный синоним:

`FullDescr`

Возвращаемое значение:

Строковое значение, содержащее полное наименование выбранного элемента справочника.

Описание:

Метод `ПолноеНаименование` возвращает строку, содержащую полное наименование выбранного элемента справочника (с наименованиями групп всех вышестоящих уровней, разделенными символом "/").

Пример:

```
ПолнИмя = Спр.Подразделения.ПолноеНаименование();
```

См. также: `НайтиПоНаименованию`

НайтиЭлемент

Найти элемент справочника по значению.

Синтаксис:

`НайтиЭлемент(<Элемент>)`

Англоязычный синоним:

`Findltem`

Параметры:

`<Элемент>`	Выражение со значением элемента справочника.

Возвращаемое значение:

Число 1 — если действие выполнено;

Число 0 — если действие не выполнено (элемент не найден).

Описание:

Метод `НайтиЭлемент` выполняет поиск элемента справочника по значению, заданному параметром `<Элемент>`, и позиционирует объект справочник на этом элементе.

Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.

Пример:

```
// НашБанк должен иметь тип «справочник.Банки»
Бнк = СоздатьОбъект("Справочник.Банки");
Бнк.ИспользоватьДату(ДатаДок);
Если НашБанк.Выбран() = 1 Тогда
   // Позиционируем созданный объект Бнк на заданном элементе
   Бнк.НайтиЭлемент(НашБанк);
```

```
   Процент = Бнк.Процент;
   НазваниеБанка = Бнк.Наименование;
КонецЕсли;
```

См. также: ТекущийЭлемент

НайтиПоКоду

Найти элемент справочника по коду.

Синтаксис:

НайтиПоКоду(<Код>, <ФлагПоиска>)

Англоязычный синоним:

FindByCode

Параметры:

<Код>	Выражение со значением искомого кода.
<ФлагПоиска>	Необязательный параметр. Числовое выражение — флаг поиска: 0 — поиск во всем справочнике вне зависимости от родителя; 1 — поиск внутри установленного подчинения (родителя); 2 — поиск по полному коду. Значение по умолчанию: 0 — для справочников, у которых код уникален во всем справочнике; 2 — для справочников, у которых код уникален в группе.

Возвращаемое значение:

Число 1 — если действие выполнено;
Число 0 — если действие не выполнено (элемент не найден).

Описание:

Метод НайтиПоКоду выполняет поиск элемента справочника по значению кода, заданному параметром <Код>, и позиционирует объект справочник на этом элементе. Если справочник многоуровневый, то полный код выбираемого элемента можно задавать, разделяя коды уровней символом "/".

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
// Контекст журнала расчетов.
// Реквизит Подразд имеет тип «справочник»
Пдр = СоздатьОбъект("Справочник.Подразделения");
Пдр.ИспользоватьДату(ДатаДок);
Если Документ.Подразд <> 0 Тогда
   // если в документе код подразделения указан...
   Пдр.НайтиПоКоду(Документ.Подразд);
   Если Пдр.Выбран() > 0 Тогда
      Сообщить("Есть такое подразделение !")
   Иначе
      Сообщить("Нет такого подразделения!")
   КонецЕсли;
КонецЕсли;
```

См. также: ПолныйКод

НайтиПоНаименованию

Найти элемент справочника по наименованию.

Синтаксис:

НайтиПоНаименованию(<Наименование>, <Режим>, <ФлагПоиска>)

Англоязычный синоним:

FindByDescr

Параметры:

<Наименование>	Строковое выражение с наименованием искомого элемента справочника.
<Режим>	Необязательный параметр. Числовое выражение — режим поиска: 1 — поиск внутри установленного подчинения (родителя); 0 — поиск во всем справочнике вне зависимости от родителя. Значение по умолчанию — 1.
<ФлагПоиска>	Необязательный параметр. Числовое выражение — флаг поиска: 1 — найти точное соответствие наименования; 0 — найти наименование по первым символам. Значение по умолчанию — 0.

Возвращаемое значение:

Число 1 — если действие выполнено;
Число 0 — если действие не выполнено (элемент не найден).

Описание:

Метод НайтиПоНаименованию выполняет поиск элемента справочника по наименованию, заданному параметром <Наименование> и позиционирует объект справочник на этом элементе.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.ИспользоватьДату(ДатаДок);
// Позиционируем созданный объект Спр по известному наименованию
Спр.НайтиПоНаименованию("Иванов", 1);
Если Спр.Выбран() > 0 Тогда
   Оклад=Спр.Оклад;
   Подразделение = Спр.Подразделение;
Иначе
   Предупреждение("Нет у нас Ивановых!");
КонецЕсли;
```

См. также: ПолноеНаименование

НайтиПоРеквизиту

Найти элемент справочника по значению реквизита.

Синтаксис:

НайтиПоРеквизиту(<ИмяРеквизита>, <Значение>, <ФлагГлобальногоПоиска>)

Англоязычный синоним:

FindByAttribute

Параметры:

<ИмяРеквизита>	Строковое выражение с наименованием реквизита.
<Значение>	Значение реквизита для поиска.
<ФлагГлобальногоПоиска>	Числовое выражение. Если 0, то поиск должен выполняться в пределах подчинения справочника, если 1, то поиск должен выполняться по всему справочнику.

Возвращаемое значение:

Число 1 — если действие выполнено;

Число 0 — если действие не выполнено (элемент не найден).

Описание:

Метод НайтиПоРеквизиту выполняет поиск первого элемента с указанным значением заданного реквизита и позиционирует объект справочник на этом элементе.

Данный метод может использоваться только в том случае, если в конфигураторе при описании данного реквизита установлен признак «Сортировка» (Свойства реквизита - Дополнительные - Сортировка).

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
Спр.ИспользоватьДату(ДатаДок);
// Позиционируем созданный объект Спр по реквизиту
Спр.НайтиПоРеквизиту("СерияПаспорта", "XXVII-ОР", 1);
Если Спр.Выбран() > 0 Тогда
   Имя = Спр.Наименование;
Иначе
   Предупреждение("Не найден сотрудник с таким паспортом!");
КонецЕсли;
```

ВыбратьЭлементы

Открыть выборку элементов справочника

Синтаксис:

ВыбратьЭлементы(<Режим>)

Англоязычный синоним:

SelectItems

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — выбирать элементы с учетом иерархии; 0 — выбирать элементы без учета иерархии. Значение по умолчанию — 1.

Возвращаемое значение:

Число 1 — если действие выполнено и в выборке есть хотя бы один элемент;

Число 0 — если действие не выполнено или в выборке нет ни одного элемента.

Описание:

Метод ВыбратьЭлементы предоставляет возможность выбирать элементы (открывает выборку) при помощи метода ПолучитьЭлемент.

Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника в порядке, установленном заранее при помощи методов: ПорядокНаименований, ПорядокКодов, ВключатьПодчиненные, ИспользоватьРодителя, ИспользоватьВладельца, ИспользоватьДату.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Акц = СоздатьОбъект("Справочник.Акционеры");
Таб = СоздатьОбъект("Таблица");
Акц.ИспользоватьДату(ДатаОтчета);
Таб.ВывестиСекцию("Отчет");
// Открываем выборку Акц.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока Акц.ПолучитьЭлемент() > 0 Цикл
   Если Акц.ЭтоГруппа() = 1 Тогда
      Таб.ВывестиСекцию("Группа");
   Иначе
      Таб.ВывестиСекцию("Акционер");
   КонецЕсли;
КонецЦикла;
```

См. также: ПолучитьЭлемент, ПорядокНаименований, ПорядокКодов, ВключатьПодчиненные, ИспользоватьРодителя, ИспользоватьВладельца, ИспользоватьДату.

ВыбратьЭлементыПоРеквизиту

Открыть выборку элементов справочника по значению реквизита.

Синтаксис:

```
ВыбратьЭлементыПоРеквизиту(<ИмяРеквизита>, <Значение>, <РежимИерархии>,
                           <РежимГрупп>)
```

Англоязычный синоним:

```
Select ItemsByAttribute
```

Параметры:

<ИмяРеквизита>	Строковое выражение с именем реквизита справочника, который задает порядок обхода элементов справочника.
<Значение>	Значение реквизита для выборки.
<РежимИерархии>	Необязательный параметр. Числовое выражение: 1 — выбирать элементы с учетом иерархии; 0 — выбирать элементы без учета иерархии. Значение по умолчанию — 1.
<РежимГрупп>	Числовое выражение: 1 — выбирать среди групп справочника; 0 — выбирать среди элементов справочника.

Возвращаемое значение:

Число; 1 — если действие выполнено и в выборке есть хотя бы один элемент; 0 — если действие не выполнено или в выборке нет ни одного элемента.

Описание:

Метод ВыбратьЭлементыПоРеквизиту предоставляет возможность выбирать элементы (открывает выборку) при помощи метода ПолучитьЭлемент.

Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника, имеющих значение реквизита <ИмяРеквизита> равным <Значение>, в порядке, установленном параметрами <РежимИерархии> и <РежимГрупп>, а также согласно установкам, сделанным заранее при помощи методов:

ПорядокНаименований,
ПорядокКодов,
ВключатьПодчиненные,
ИспользоватьРодителя,
ИспользоватьВладельца,
ИспользоватьДату.

Данный метод может использоваться только в том случае, если в конфигураторе при описании данного реквизита установлен признак «Сортировка» (Свойства реквизита - Дополнительные - Сортировка).

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Акц = СоздатьОбъект("Справочник.Акционеры");
Таб = СоздатьОбъект("Таблица");
Акц.ИспользоватьДату(ДатаОтчета);
```

```
Таб.ВывестиСекцию("Отчет");
// Открываем выборку
Акц.ВыбратьЭлементыПоРеквизиту("Отдел", НомерОтдела, 1, 0);
// Цикл получения элементов справочника
Пока Акц.ПолучитьЭлемент() > 0 Цикл
   Если Акц.ЭтоГруппа() = 1 Тогда
      Таб.ВывестиСекцию("Группа");
      Таб.ВывестиСекцию("Акционер");
   КонецЕсли;
КонецЦикла;
```

См. также: ПолучитьЭлемент, ПорядокНаименований, ПорядокКодов, ОбратныйПорядок, ВключатьПодчиненные, ИспользоватьРодителя, ИспользоватьВладельца, ИспользоватьДату

ОбратныйПорядок

Установить порядок выборки элементов справочника.

Синтаксис:

ОбратныйПорядок(<Режим>)

Англоязычный синоним:

BackwardOrder

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — выбирать элементы справочника в обратном порядке; 0 — выбирать элементы справочника в прямом порядке. Значение по умолчанию — 1.

Возвращаемое значение:

Текущее значение порядка выборки элементов справочника (на момент до исполнения метода): 1 — обратный порядок выборки элементов справочника; 0 — выборка элементов справочника в прямом порядке.

Описание:

Метод ОбратныйПорядок устанавливает порядок выборки элементов справочника. Данный метод обычно используется до вызова одного из методов: ВыбратьЭлементы, ВыбратьЭлементыПоРеквизиту который фактически открывает выборку. Дальнейшая выборка при помощи ПолучитьЭлемент будет происходить в заданном порядке выборки.

По умолчанию, выборка элементов справочника выполняется в прямом порядке. Поэтому реально имеет смысл применять данный метод только в том случае, если надо получить обратный порядок выборки.

В тексте программного модуля возможно использование данного метода как процедуры или как функции. При использовании в качестве функции, возвращаемое значение соответствует текущему порядку выборки, которое было до вызова данного метода.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Акц = СоздатьОбъект("Справочник.Акционеры");
Акц.ИспользоватьДату(ДатаОтчета);
Таб.ВывестиСекцию("Отчет");
// Открываем выборку Акц.ВыбратьЭлементы();
Акц.ПорядокКодов();
Акц.ОбратныйПорядок(1);
// Цикл получения элементов справочника.
Пока Акц.ПолучитьЭлемент() > 0 Цикл
   Если Акц.Это Группа() = 1 Тогда
      Таб.ВывестиСекцию("Группа");
   Иначе
      Таб.ВывестиСекцию("Акционер");
   КонецЕсли;
КонецЦикла;
```

См. также: ВыбратьЭлементы, ПолучитьЭлемент, ПорядокНаименований, ПорядокКодов, ВключатьПодчиненные, ИспользоватьРодителя, ИспользоватьВладельца, ИспользоватьДату

ПолучитьЭлемент

Получить из выборки следующий элемент справочника.

Синтаксис:

ПолучитьЭлемент(<Режим>)

Англоязычный синоним:

GetItem

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 1 — надо включать в выборку все подчиненные элементы, если 0 — не надо включать подчиненные элементы. Значение по умолчанию — 1.

Возвращаемое значение:

Числовое значение: 1 — если элемент выбран успешно, 0 — если элемент не выбран (отсутствует).

Описание:

Метод ПолучитьЭлемент выбирает следующий элемент справочника в последовательности выборки, открытой перед этим при помощи метода ВыбратьЭлементы.

Данный метод может применяться только для объектов, созданных функцией СоздатьОбъект, и используется для организации цикла поиска по справочнику.

Пример:

```
Спр = СоздатьОбъект("Справочник.ОсновныеСредства");
// Задаем выборку в порядке кодов
Спр.ПорядокКодов();
// Открываем выборку
Спр.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока Спр.ПолучитьЭлемент() > 0 Цикл
   Сообщить("===" + Спр.Наименование);
КонецЦикла;
```

См. также: ВыбратьЭлементы, ПолучитьЭлемент, ПорядокНаименований, ОбратныйПорядок, ПорядокКодов, ВключатьПодчиненные, ИспользоватьРодителя, ИспользоватьВладельца, ИспользоватьДату

ИспользоватьДату

Установить дату выборки периодических реквизитов справочника.

Синтаксис:

ИспользоватьДату[<Дата>, <УстСразу>)

Англоязычный синоним:

UseDate

Параметры:

<Дата> Необязательный параметр. Выражение со значением типа «дата».

<УстСразу> Необязательный параметр. Число: если 1, то дата, переданная в качестве параметра, будет установлена уже в текущей выборке; если 0 — то дата, переданная в качестве параметра будет установлена при следующей выборке. Значение по умолчанию — 0. Например:

если "Цена" — периодический реквизит справочника, то

```
Спр.ИспользоватьДату(Д1, 1);
Спр.ВыбратьЭлементы();
А = Спр.Цена; — возвращает значение на дату Д1
Спр.ИспользоватьДату(Д2, 1);
Б = Спр.Цена; — возвращает значение на дату Д2!
```

Если <УстСразу> — 0 (или опущен), то

```
Спр.ИспользоватьДату(Д1);
Спр.ВыбратьЭлементы();
А = Спр.Цена; — возвращает значение на дату Д1
Спр.ИспользоватьДату(Д2);
Б=Спр.Цена; — тоже возвращает значение на дату Д1!
Спр.ВыбратьЭлементы();
В=Спр.Цена; — только теперь возвращает значение на Д2.
```

Возвращаемое значение:

Текущее значение используемой даты (на момент до исполнения метода).

Описание:

Метод ИспользоватьДату устанавливает для объекта типа «справочник» дату, на которую будут в дальнейшем выбираться (или записываться) значения периодических реквизитов справочника.

Это очень важный метод, о котором необходимо всегда помнить при работе со справочниками, имеющими периодические реквизиты. Если он пропущен, то чначения выбранных периодических реквизитов справочника будут не определены.

Замечание: Если к объекту типа «справочник» однажды применен метод ИспользоватьДату, то в дальнейшем, чтобы выбирать значения периодических реквизитов справочника, нельзя применять к этому же объекту методы Получить и Установить, т. е. в такой последовательности эти методы несовместимы.

Замечание: Данный метод имеет следующую особенность применения: его нельзя использовать «через две точки». Например, если в документе с именем "ДокНакл" есть реквизит "Фирма" типа «справочник», у которого есть периодические реквизиты НДС и СН, то следующий оператор:

```
ДокНакл.Фирма.ИспользоватьДату(Дата);
```

работать не будет. В данном случае следует просто использовать промежуточную переменную, например:

```
ФирДок = ДокНакл.Фирма;
ФирДок.ИспользоватьДату(РабочаяДата());
НДСДок = ФирДок.НДС;
```

Пример:

```
// Здесь мы работаем в локальном контексте модуля Формы
// редактирования документа.
// Реквизит "Должность" в документе является справочником,
// у которого "МинОкл" - периодический реквизит
Длж = СоздатьОбъект("Справочник.Должности");
Длж.ИспользоватьДату(ДатаДок);
Длж.НайтиЭлемент(Должность);
Сообщить(Длж.Наименование + " Минимальный оклад" + Строка(Длж.МинОкл));
```

См. также: Получить, Установить, НайтиЭлемент, ВыбратьЭлементы

ИспользоватьВладельца

Установить выборку подчиненного справочника.

Синтаксис:

```
ИспользоватьВладельца(<Владелец>, <ФлагИзменения>)
```

Англоязычный синоним:

```
UseOwner
```

Параметры:

<Владелец>	Необязательный параметр. Выражение со значением элемента справочника-владельца.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения владельца. 1 — пользователь может изменить владельца интерактивно, 0 — пользователь не может интерактивно изменить владельца. Этот параметр используется в случае использовсния данного метода для объектов типа «справочник», которые являются реквизитами формы или реквизитами диалога.

Возвращаемое значение:

Значение элемента справочника-владельца для текущего подчиненного справочника (на момент до исполнения метода).

Описание:

Метод ИспользоватьВладельца может применяться к объектам типа «справочник» в двух случаях:

- Для объектов, созданных функцией СоздатьОбъект, метод ИспользоватьВладельца устанавливает элемент справочника-владельца (которому подчинен текущий подчиненный справочник) в качестве параметра выборки. Данный метод используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить только среди тех элементов текущего подчиненного справочника, для которых владельцем является заданное значение элемента справочника-владельца <Владелец>. При записи нового элемента текущего справочника данный метод также задает владельца для нового элемента.
- Для объектов типа «справочник», которые являются реквизитами формы (например, в форме документа — реквизит документа типа «справочник») или реквизитами диалога (например, в форме отчета — реквизит диалога типа «справочник») метод ИспользоватьВладельца позволяет программно установить некоторое значение справочника-владельца в качестве владельца, который будет использован при интерактивном выборе значения данного реквизита.

Пример:

```
// Процедура выводит список детей сотрудника
// Справочник детей подчинен справочнику сотрудников
Процедура ВыводСпискаДетей(Сотр)
   СпрД = СоздатьОбъект("Справочник.Дети");
      // в качестве параметра функции ИспользоватьВладельца
      // передаем параметр, переданный в процедуру
   СпрД.ИспользоватьВладельца(Сотр);
   Сообщить("Дети сотрудника " + Сотр.Наименование);
   СпрД.ВыбратьЭлементы();
   Пока СпрД.ПолучитьЭлемент() > 0 Цикл
```

```
      Сообщить(СпрД.Наименование);
   КонецЦикла;
КонецПроцедуры
```

Пример:

```
// В форме элемента справочника Товары есть реквизит Единица,
// который имеет тип подчиненного справочника «Единицы»
// здесь мы можем задать ему в качестве владельца текущий элемент:
Единица.ИспользоватьВладельца(ТекущийЭлемент());
```

См. также: ВыбратьЭлементы, ПолучитьЭлемент, СоздатьОбъект

ИспользоватьРодителя

Установить выборку по группе справочника.

Синтаксис:

ИспользоватьРодителя(<Группа>)

Англоязычный синоним:

UseParent

Параметры:

<Группа>	Необязательный параметр. Выражение со значением группы справочника.

Возвращаемое значение:

Значение текущей группы для справочника (на момент до исполнения метода).

Описание:

Метод ИспользоватьРодителя устанавливает группу текущего справочника в качестве параметра выборки.

Данный метод используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить только среди элементов текущего справочника, принадлежащих указанной группе.

При добавлении нового элемента текущего справочника данная установка также будет являться свойством нового элемента.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
// Это процедура формирования отчета – "Список сотрудников"
// ВыбСотр – реквизит диалога типа «справочник.Сотрудники»
Процедура ВыводСпискаСотрудников()
   Таб = СоздатьОбъект("Таблица");
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   Спр.ИспользоватьДату(ДатаОтчета);
   Если ВыбСотр.Выбран() = 0 Тогда
      // если сотрудник в диалоге не выбран,
      // то формируем отчет без условий
      Заг = "По всем сотрудникам";
   ИначеЕсли ВыбСотр.ЭтоГруппа() = 1 Тогда
      // если в диалоге выбрана группа сотрудников, то
      // задаем выборку элементов справочника из одной группы
      Спр.ИспользоватьРодителя(ВыбСотр);
      // Задаем выборку всех подчиненных элементов справочника
      Заг = "По сотрудникам группы " + ВыбСотр.Наименование;
   Иначе
      // если в диалоге выбран сотрудник, то
      // формируем отчет только по нему
      Спр.НайтиЭлемент(ВыбСотр);
      Заг = "По сотруднику " + ВыбСотр.Наименование;
      Таб.ВывестиСекцию("Отчет");
      Таб.ВывестиСекцию("Сотрудник");
      Перейти ~МЕТ;
   КонецЕсли;
   ЧислоСтрок = 0;
   Таб.ВывестиСекцию("Отчет");
   // Открываем выборку Спр.ВыбратьЭлементы();
   Пока Спр.ПолучитьЭлемент() > 0 Цикл
      ЧислоСтрок = ЧислоСтрок + 1;
      Если Спр.ЭтоГруппа() = 1 Тогда
         Таб.ВывестиСекцию("Группа");
```

```
      Иначе
         Таб.ВывестиСекцию("Сотрудник");
      КонецЕсли;
      Состояние("В отчет выведено "+ЧислоСтрок+" строк.");
   КонецЦикла;
~МЕТ:
   //Вызов выходного отчета в окно просмотра и редактирования.
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 4, 0);
   Таб.Показать("Список товаров по каталогу", "");
   ВыбТовар = 0;
КонецПроцедуры
```

См. также: ВыбратьЭлементы, ПолучитьЭлемент, СоздатьОбъект

ВключатьПодчиненные

Установить флаг выборки всех подчиненных элементов.

Синтаксис:

```
ВключатьПодчиненные(<Режим>]
```

Англоязычный синоним:

```
IncludeChildren
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: если 1 — надо включать в выборку все подчиненные элементы, если 0 — не надо включать подчиненные элементы.

Возвращаемое значение:

Текущее числовое значение режима выборки подчиненных элементов справочника (на момент до исполнения метода).

Описание:

Метод ВключатьПодчиненные устанавливает флаг выборки всех подчиненных элементов (то есть раскручивания дерева справочника глубже текущего уровня).

Данный метод используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника в соответствии с установленным режимом выборки <Режим>. По умолчанию в выборку всегда включаются подчиненные элементы, поэтому реально имеет смысл применять данный метод только в том случае, если надо отменить включение подчиненных.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
// Это процедура формирования отчета - "Список Товаров"
// ВыбТовар - реквизит диалога типа «справочник.Товар», которым
// задается группа товаров для отображения.
//В процедуру передается параметр Режим, которым задается,
// отображать или нет вложенные подгруппы товаров выбранной группы
Процедура ВыводСпискаТоваров(Режим)
   Таб = СоздатьОбъект("Таблица");
   Тов = СоздатьОбъект("Справочник.Товары");
   Тов.ИспользоватьДату(ДатаОтчета);
   Если ВыбТовар.Выбран() = 0 Тогда
      //если товар в диалоге не выбран, то формируем отчет без условий
      Заг = "По всем товарам.";
   ИначеЕсли ВыбТовар.ЭтоГруппа() = 1 Тогда
      // если в диалоге выбрана группа товаров, то
      // Задаем выборку элементов Справочника из одной группы
      Тов.ИспользоватьРодителя(ВыбТовар);
      Заг = "По товарам группы " + ВыбТовар.Наименование;
      // Задаем выборку всех подчиненных элементов Справочника
      Тов.ВключатьПодчиненные(Режим);
   Иначе
      // если в диалоге выбран товар, то
      // формируем отчет только по этому товару
      Тов.НайтиЭлемент(ВыбТовар);
      Заг = "По товару " + ВыбТовар.Наименование;
      Таб.ВывестиСекцию("Отчет");
      Таб.ВывестиСекцию("Товар");
```

```
      Перейти ~МЕТ;
   КонецЕсли;
   ЧислоСтрок = 0;
   Таб.ВывестиСекцию("Отчет");
   // Открываем выборку
   Тов.ВыбратьЭлементы();
   Пока Тов.ПолучитьЭлемент() > 0 Цикл
      ЧислоСтрок = ЧислоСтрок + 1;
      Если Тов.ЭтоГруппа() = 1 Тогда
         Таб.ВывестиСекцию("Группа");
      Иначе
         Таб.ВывестиСекцию("Товар");
      КонецЕсли;
      Состояние("В отчет выведено " + ЧислоСтрок + " строк.");
   КонецЦикла;
   //Вызов выходного отчета в окно просмотра и редактирования.
~МЕТ:
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 4, 0);
   Таб.Показать("Список товаров по каталогу", "");
КонецПроцедуры
```

См. также: ВыбратьЭлементы, ПолучитьЭлемент, СоздатьОбъект

ПорядокКодов

Установить порядок выборки по возрастанию кода.

Синтаксис:

```
ПорядокКодов()
```

Англоязычный синоним:

```
OrderByCode
```

Описание:

Метод ПорядокКодов устанавливает режим выборки элементов справочника в порядке возрастания кодов элементов.

Данный метод обычно используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника в порядке возрастания кодов элементов.

По умолчанию выборка элементов справочника производится в порядке основного представления справочника, которое задается в конфигураторе. Поэтому реально имеет смысл применять данный метод только в том случае, если основное представление справочника — наименование, а надо получить элементы в порядке кодов.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
// Задаем выборку в порядке кодов Спр.ПорядокКодов();
// Открываем выборку Спр.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока (Спр.ПолучитьЭлемент() > 0) Цикл
   Состояние(Спр.Наименование);
КонецЦикла;
```

ПорядокНаименований

Установить порядок выборки по возрастанию наименования.

Синтаксис:

```
ПорядокНаименований()
```

Англоязычный синоним:

```
OrderByDescr
```

Описание:

Метод ПорядокНаименований устанавливает режим выборки элементов справочника в порядке возрастания наименования элементов.

Данный метод обычно используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника в порядке возрастания наименования элементов.

По умолчанию выборка элементов справочника производится в порядке основного представления справочника, которое задается в конфигураторе. Поэтому реально имеет смысл применять данный метод только в том случае, если основное представление справочника — код, а надо получить элементы в порядке наименований.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
// Задаем выборку в порядке наименований
Спр.ПорядокНаименований();
// Открываем выборку
Спр.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока (Спр.ПолучитьЭлемент() > 0) Цикл
   Состояние(Спр.Наименование);
КонецЦикла;
```

ПорядокРеквизита

Установить порядок выборки по возрастанию значения реквизита.

Синтаксис:

```
ПорядокРеквизита(<ИмяРеквизита>)
```

Англоязычный синоним:

```
OrderByAttribute
```

Параметры:

<ИмяРеквизита>	Строковое выражение с именем реквизита справочника, который задает порядок обхода элементов справочника.

Описание:

Метод ПорядокРеквизита устанавливает режим выборки элементов справочника в порядке возрастания значения указанного реквизита <ИмяРеквизита>.

Данный метод обычно используется до вызова метода ВыбратьЭлементы, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьЭлемент будет происходить среди элементов текущего справочника в порядке возрастания значения реквизита.

Данный метод может использоваться только в том случае, если в конфигураторе при описании данного реквизита установлен признак «Сортировка» (Свойства реквизита - Дополнительные - Сортировка).

Данный метод может использоваться только для объектов, созданных Функцией СоздатьОбъект.

Пример:

```
Спр = СоздатьОбъект("Справочник.Сотрудники");
// Задаем выборку в порядке возрастания окладов
Спр.ПорядокРеквизита("Оклад");
// Открываем выборку
Спр.ВыбратьЭлементы();
// Цикл получения элементов справочника
Пока Спр.ПолучитьЭлемент() = 1 Цикл
   Состояние(Спр.Наименование + " - " + Спр.Оклад);
КонецЦикла;
```

Новый

Добавить новый элемент справочника.

Синтаксис:

```
Новый()
```

Англоязычный синоним:

```
New
```

Описание:

Метод Новый инициализирует создание нового элемента справочника. Собственно запись нового элемента происходит при вызове метода Записать. После инициализации создания нового элемента справочника, как правило, производится заполнение его реквизитов с последующим вызовом метода Записать.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Процедура ВводСотрудника()
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   // добавляем новый элемент справочника
   Спр.Новый();
   // Определяем реквизиты нового элемента справочника
```

```
   Спр.Наименование = ФИО;
   Спр.Код = ТН;
   Спр.Оклад = Оклад;
   Спр.Аванс = Аванс;
   // ...
   // Записываем новый элемент справочника
   Спр.Записать();
КонецПроцедуры
```

НоваяГруппа

Добавить новую группу справочника.

Синтаксис:

```
НоваяГруппа()
```

Англоязычный синоним:

```
NewGroup
```

Описание:

Метод НоваяГруппа добавляет пустую запись новой группы в справочник. Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Процедура ВводГруппыСотрудников()
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   // добавляем новую группу справочника
   Спр.НоваяГруппа();
   // Определяем реквизиты новой группы справочника
   Спр.Наименование = "Расчетчик3";
   Спр.Код = 3;
   // Записываем на диск
   Спр.Записать();
КонецПроцедуры
```

ПрефиксКода

Установить для справочника текущий префикс кода.

Синтаксис:

```
ПрефиксКода(<Префикс>)
```

Англоязычный синоним:

```
CodePrefix
```

Параметры:

<Префикс>	Необязательный параметр. Строковое выражение — новый префикс кодов элементов справочника.

Возвращаемое значение:

Строковое значение текущего префикса кодов элементов справочника (на момент до исполнения метода).

Описание:

Метод ПрефиксКода устанавливает новый текущий префикс для автоматического создания кодов элементов справочника.

Пример:

```
ПрефиксКода("01-");
```

См. также: ПрефиксАвтоНумерации, УстановитьНовыйКод

УстановитьНовыйКод

Установить новый код с префиксом кода для справочника.

Синтаксис:

```
УстановитьНовыйКод(<Префикс>)
```

Англоязычный синоним:

```
SetNewCode
```

Параметры:

<Префикс>	Строковое выражение — префикс кода элемента справочника.

Описание:

Метод УстановитьНовыйКод устанавливает новый код элемента справочника с префиксом <Префикс>.

Пример:

```
УстановитьНовыйКод("01-");
```

См. также: ПрефиксАвтоНумерации, ПрефиксКода

НазначитьТип

Назначить тип для реквизита неопределенного вида.

Синтаксис:

```
НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
SetType
```

Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита справочника неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту справочника. Например: "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных. Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:

Метод НазначитьТип позволяет назначить тип для реквизита, которому в конфигураторе назначен тип «Неопределенный».

Пример:

```
Номенклатура.НазначитьТип("ТМЦ", "Справочник.Товары");
```

Записать

Записать (обновить) элемент справочника.

Синтаксис:

```
Записать()
```

Англоязычный синоним:

```
Write
```

Описание:

Метод Записать выполняет запись (обновление) элемента справочника.

Замечание. Если этот метод применяется в Модуле формы элемента справочника непосредственно к элементу справочника локального контекста, то данный метод отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Записать".

Пример:

```
Процедура ВводСотрудника()
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   // добавляем новый элемент справочника
   Спр.Новый();
   // Определяем реквизиты нового элемента справочника
   Спр.Наименование = ФИО;
   Спр.Код = ТН;
   Спр.Оклад = Оклад;
   Спр.Аванс = Аванс;
   // ...
   // Записываем новый элемент справочника
   Спр.Записать();
КонецПроцедуры
```

Удалить

Удалить элемент справочника.

Синтаксис:

```
Удалить(<Режим>)
```

Англоязычный синоним:

```
Delete
```

Параметры:

<Режим>	Числовое выражение: 1 — непосредственное удаление; 0 — пометка на удаление. Необязательный параметр. Значение по умолчанию — 1.

Описание:

Метод Удалить удаляет (или делает пометку на удаление) текущий элемент или группу справочника.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Замечание: Непосредственное удаление объекта следует применять очень аккуратно, так как это действие может нарушить ссылочную целостность информации. Данный режим не рекомендуется использовать, если на данный объект могут быть ссылки в других объектах, например в реквизитах существующих документов.

Пример:

```
Процедура УдалениеСотрудника(Сотр)
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   Если Сотр.Выбран() > 0 Тогда
      // позиционируем справочник на заданном элементе
      Спр.НайтиЭлемент(Сотр);
      Если Спр.Выбран() > 0 Тогда
         Если Вопрос("Удалять??? ", 1) = 1 Тогда
            // удаляем элемент справочника
            Спр.Удалить(1);
         КонецЕсли;
      Иначе
         Предупреждение("Некого удалять!");
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: ПометкаУдаления, СнятьПометкуУдаления

Блокировка

Установить/прочитать режим блокировки.

Синтаксис:

```
Блокировка(<ВклВыкл>)
```

Англоязычный синоним:

```
Locking
```

Параметры:

<ВклВыкл>	Необязательный параметр. Число: 1 — включить; 0 — выключить.

Возвращаемое значение:

Если при вызове метода параметр <ВклВыкл> не задан, то возвращается режим блокировки до выполнения метода. Число: 1 — заблокирован; 0 — свободен.

Если при вызове метода параметр <ВклВыкл> задан, то возвращается результат выполнения метода блокировки. Число: 1 — успешно; 0 — не получилось.

Описание:

Метод Блокировка позволяет установить/прочитать режим блокировки. Пример:

```
Блокировка(0);
```

ПометкаУдаления

Прочитать флаг пометки удаления элемента справочника.

Синтаксис:

```
ПометкаУдаления()
```

Англоязычный синоним:

```
DeleteMark
```

Возвращаемое значение:

Числовое значение: 1 — если на элементе справочника стоит пометка удаления; 0 — если нет пометки удаления.

Описание:

Метод ПометкаУдаления позволяет прочитать значение пометки удаления текущего элемента справочника.

Пример:

```
Процедура УсловноеУдалениеСотрудника(Сотр)
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   Если Сотр.Выбран() > 0 Тогда
      // позиционируем справочник на заданном элементе
      Спр.НайтиЭлемент(Сотр);
      Если Спр.Выбран() > 0 Тогда
         Если Вопрос("Удалять??? ", 1) = 1 Тогда
```

```
            Если Спр.ПометкаУдаления() = 1 Тогда
               Предупреждение("Уже помечен на удаление!");
            Иначе
               // удаляем элемент справочника
               Спр.Удалить(0);
            КонецЕсли;
         КонецЕсли;
      Иначе
         Предупреждение("Некого удалять!");
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: Удалить, СнятьПометкуУдаления

СнятьПометкуУдаления

Снять пометку удаления элемента справочника.

Синтаксис:

```
СнятьПометкуУдаления()
```

Англоязычный синоним:

```
ClearDeleteMark
```

Описание:

Метод СнятьПометкуУдаления снимает пометку удаления текущего элемента справочника.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Процедура ВосстановлениеУдаленныхСотр(Сотр)
   Спр = СоздатьОбъект("Справочник.Сотрудники");
   Если Сотр.Выбран() > 0 Тогда
      // позиционируем справочник на заданном элементе
      Спр.НайтиЭлемент(Сотр);
      Если Спр.Выбран() > 0 Тогда
         Если Спр.ПометкаУдаления() = 1 Тогда
            // восстанавливаем элемент справочника
            Спр.СнятьПометкуУдаления();
         КонецЕсли;
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: Удалить, ПометкаУдаления

Методы контекста Модуля формы элемента справочника

Описанные в данном разделе методы доступны только в контексте Модуля формы элемента справочника (см. «Виды программных модулей»).

Модифицированность

Возвратить признак изменения реквизитов в текущей форме элемента справочника.

Синтаксис:

```
Модифицированность()
```

Англоязычный синоним:

```
Modify
```

Возвращаемое значение:

Числовое значение: 1 — если реквизиты текущей формы элемента справочника были изменены; 0 — если нет.

Описание:

Метод Модифицированность возвращает признак изменения (в режиме исполнения он отображается символом (*) в заголовке окна формы).

Замечание. Данный метод доступен только в контексте Модуля формы элемента справочника (см. «Виды программных модулей»). Действие данного метода относится только к текущему элементу справочника, который доступен в локальном контексте Модуля формы элемента справочника.

Пример:

ИспользоватьДату

Установить дату, на которую будут записываться периодические реквизиты элемента справочника, форма которого открыта.

Синтаксис:

```
ИспользоватьДату(<Дата>, <Обновить>)
```

Англоязычный синоним:

```
UseDate
```

Параметры:

<Дата>	Выражение со значением типа «дата». Дата, на которую будут записываться периодические реквизиты элемента справочника, форма которого открыта.
<Обновить>	Необязательный параметр. Число: 1 — обновить периодические реквизиты формы на заданную дату; 0 — не обновлять периодические реквизиты формы. Значение по умолчанию — 0;

Возвращаемое значение:

Значение используемой даты (на момент до исполнения метода).

Описание:

Метод ИспользоватьДату устанавливает для элемента справочника дату, на которую будут записываться периодические реквизиты.

Если данный метод опущен, то значения выбранных периодических реквизитов справочника будут записываться на текущую рабочую дату.

Замечание. Данный метод доступен только в контексте Модуля формы элемента справочника (см. «Виды программных модулей»). Действие данного метода относится только к текущему элементу справочника, который доступен в локальном контексте Модуля формы элемента справочника.

Пример:

```
// Здесь мы работаем в локальном контексте модуля Формы
// элемента справочника.
// в справочнике есть несколько реквизитов,
// которые являются периодическими реквизитами.
// В форме существует элемент диалога ВыбДата типа «Дата»
Если Число(ВыбДата) <> 0 Тогда
   // задано некоторое значение ВыбДата
   ИспользоватьДату(ВыбДата);
КонецЕсли;
```

СохранениеПериодическихРеквизитов

Установить режим отображения диалога сохранения периодических реквизитов при записи элемента справочника.

Синтаксис:

```
СохранениеПериодическихРеквизитов(<ВариантВызова>, <Список>)
```

Англоязычный синоним:

```
PerIodicAttributesSaving
```

Параметры:

<ВариантВызова>	Число, определяет вариант вызова метода: 0 — отключить автоматический режим определения списка измененных реквизитов и показ диалога (параметр <Список> не используется); 1 — включить автоматический режим определения списка измененных реквизитов с показом диалога (параметр <Список> определяет набор анализируемых реквизитов); 2 — включить автоматический режим определения списка измененных реквизитов без показа диалога (параметр <Список> определяет набор анализируемых реквизитов); 3 — выполнить определение списка измененных реквизитов с показом диалога (параметр <Список> определяет набор анализируемых реквизитов); 4 — выполнить определение списка измененных реквизитов без показа диалога (параметр <Список> определяет набор анализируемых реквизитов); 5 — выдать/установить список сохраняемых периодических реквизитов. Метод возвращает установленный список периодических реквизитов, которые будут записываться на момент до вызова метода. Если параметр <Список> указан, то устанавливается список периодических реквизитов, которые должны быть записаны.
<Список>	Необязательный параметр. Строковое выражение, в котором через запятую перечислены имена периодических реквизитов справочника, которые должны выводится в диалоге (если диалог показывается), или по которым будет записан элемент истории (если диалог

не показывается). По умолчанию предполагается, что в списке все периодические реквизиты справочника. Если указан символ "*", то считается, что в список включены все периодические реквизиты справочника.

Описание:

Метод СохранениеПериодическихРеквизитов управляет режимом ав-томатического определения списка измененных периодических реквизитов, показом диалога для выбора записываемых периодических реквизитов, а также предоставляет доступ непосредственно к списку записываемых периодических реквизитов.

Если данный метод опущен, то при записи элемента справочника диалог отображается со всеми периодическими реквизитами.

Замечание. Данный метод доступен только в контексте Модуля формы элемента справочника (см. «Виды программных модулей»). Действие данного метода относится только к текущему элементу справочника, который доступен в локальном контексте Модуля формы элемента справочника.

Пример:

```
// Здесь мы работаем в локальном контексте модуля Формы
// элемента справочника.
// в справочнике есть несколько реквизитов,
// которые являются периодическими реквизитами.
СохранениеПериодическихРеквизитов(1, "ЦенаРозн, ЦенаОптов");
```

ПросмотрИстории

Установить список периодических реквизитов, просмотр истории которых допускается.

Синтаксис:

```
ПросмотрИстории(<СписокРеквизитов>)
```

Англоязычный синоним:

```
ViewHistory
```

Параметры:

<СписокРеквизитов>	Строка со списком идентификаторов (через запятую) тех реквизитов, просмотр истории которых допускается.

Возвращаемое значение:

Текущий (на момент до вызова метода) список реквизитов, просмотр истории которых допускается просмор истории.

Описание:

Метод ПросмотрИстории позволяет установить список реквизитов для которых допускается просмор истории .

Замечание. Данный метод доступен только в контексте Модуля формы элемента справочника (см. «Виды программных модулей»). Действие данного метода относится только к текущему элементу справочника, который доступен в локальном контексте Модуля формы элемента справочника.

Пример:

```
Процедура ПриОткрытии()
   ПросмотрИстории;"Оклад, Тариф, Подразделение");
КонецПроцедуры
```

Предопределенные процедуры Модуля формы справочника

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях форм справочника (см. «Виды программных модулей»). К ним относятся программные модули: Модуль формы элемента справочника, Модуль формы группы справочника.

В основном данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Системные предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ВводНового

Предопределенная процедура при вводе нового элемента справочника.

Синтаксис:

```
ВводНового(<ПризнКопирования>, <ОбъектКопирования>)
```

Англоязычный синоним:
InputNew
Параметры:

<ПризнКопирования>	Признак того, что объект введен копированием. Число: 1 — объект введен копированием, 0 — просто новый объект. Данный признак может быть использован для анализа необходимости инициализации реквизитов нового объекта.
<ОбъектКопирования>	Объект, который был скопирован.

Описание:

Вызов процедуры ВводНового производится в системе 1С:Предприятие неявно в момент выбора пункта меню "Действия" — "Новый" при работе со справочниками. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов нового элемента справочника. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя вводить новые элементы), ввода нового элемента и открытие его формы не будет выполнено.

Процедуру ВводНового контекста справочников можно размещать в следующих программных модулях: Модуль формы элемента справочника, Модуль формы группы справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНового();
   Оклад = 1000000;
   Пдр = СоздатьОбъект("Справочник.Подразделения");
   Пдр.НайтиПоКоду(1);
   Подразделение = Пдр.ТекукщийЭлемент();
   Город = Константа.НашГород;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура при записи элемента справочника.

Синтаксис:
ПриЗаписи(<СписокПериодРекв>)
Англоязычный синоним:
OnWrite
Параметры:

<СписокПериодРекв>	Строковое значение — список разделенных запятыми изменяемых периодических реквизитов справочника. В данный параметр система 1С:Предприятие передает перечень периодических реквизитов, которые были интерактивно выбраны пользователем для обновления в окне диалога выбора. В теле процедуры значение данного параметра может быть изменено, что позволяет в данной процедуре непосредственно управлять списком записываемых значений периодических реквизитов.

Описание:

Вызов предопределенной процедуры ПриЗаписи производится системой 1С:Предприятие при интерактивной записи элемента справочника. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять некоторые элементы справочника), то запись элемента справочника не будет выполнена.

Данная предопределенная процедура может располагаться в следующих программных модулях: Модуль формы элемента справочника, Модуль формы группы справочника, Модуль формы списка справочника (см. Виды программных модулей).

Пример:

```
Процедура ПриЗаписи (СписокРекв)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если ТекущийЭлемент() = Константа.НашаФирма Тогда
         Предупреждение("У вас нет права менять реквизиты!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Методы контекста Модуля формы списка справочника

Описанные в данном разделе методы доступны только в контексте Модуля формы списка справочника (см. «Виды программных модулей»).

ИспользоватьДату

Установить дату выборки периодических реквизитов формы списка справочника.

Синтаксис:

```
ИспользоватьДату(<Дата>)
```

Англоязычный синоним:

```
UseDate
```

Параметры:

<Дата>	Выражение со значением типа «дата»

Возвращаемое значение:

Значение даты выборки периодических реквизитов формы списка справочника.

Описание:

Метод ИспользоватьДату устанавливает для формы списка справочника дату, на которую будут в дальнейшем выбираться (или записываться) значения периодических реквизитов справочника.

Данный метод доступен только в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится только к текущему справочнику, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
ИспользоватьДату(ДатаДок);
```

ИспользоватьВладельца

Установить владельца для формы списка подчиненного справочника.

Синтаксис:

```
ИспользоватьВладельца(<Владелец>, <ФлагИзменения>)
```

Англоязычный синоним:

```
UseOwner
```

Параметры:

<Владелец>	Необязательный параметр. Выражение со значением элемента справочника — нового владельца. Если параметр не задан, то значение владельца не меняется.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения владельца, 1 — пользователь может изменить владельца интерактивно, 0 — пользователь не может интерактивно изменить владельца. Если параметр не задан, то значение флага не меняется.

Возвращаемое значение:

Значение владельца для формы списка подчиненного справочника (до применения метода).

Описание:

Метод ИспользоватьВладельца устанавливает элемент связанного справочника (которому подчинен текущий справочник) в качестве владельца для формы списка подчиненного справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему подчиненному справочнику, который доступен в локальном контексте Модуля формы списка справочника.

При добавлении нового элемента текущего справочника данный параметр также будет являться свойством нового элемента.

Пример:

```
ИспользоватьВладельца(Сотр, 0);
```

ИспользоватьРодителя

Установить родителя для формы списка справочника.

Синтаксис:

```
ИспользоватьРодителя(<Родитель>, <ФлагИзменения>)
```

Англоязычный синоним:

```
UseParent
```

Параметры:

<Родитель>	Необязательный параметр. Выражение со значением группы справочника- нового родителя. Если параметр не задан, то значение родителя не меняется.
<ФлагИзменения >	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения родителя. 1 — пользователь может изменить родителя интерактивно, 0 — пользователь не может интерактивно изменить родителя. Если параметр не задан, то значение флага не меняется.

Возвращаемое значение:

Значение родителя для формы списка справочника (до применения метода).

Описание:

Метод ИспользоватьРодителя устанавливает группу справочника в качестве родителя для формы списка справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему справочнику, который доступен в локальном контексте Модуля формы списка справочника.

При добавлении нового элемента текущего справочника данный параметр также будет являться свойством нового элемента.

Пример:

```
ИспользоватьРодителя(ВыбГруппаСотр, 1);
```

ИерархическийСписок

Установить режим иерархического списка справочника.

Синтаксис:

```
ИерархическийСписок(<ФлагИерархСписка>, <ФлагИзменения>)
```

Англоязычный синоним:

```
HierarchicalList
```

Параметры:

<ФлагИерархСписка>	Флаг иерархического списка. 1 — иерархический список; 0 — неиерархический список.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения флага иерархического списка. 1 — пользователь может изменить иерархичность интерактивно, 0 — пользователь не может интерактивно изменить иерархичность.

Возвращаемое значение:

Значение флага иерархического списка для формы списка справочника (до применения метода).

Описание:

Метод ИерархическийСписок устанавливает режим иерархического списка справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
ИерархическийСписок(1, 1);
```

ВыборГруппы

Установить режим выборки групп.

Синтаксис:

```
ВыборГруппы(<Режим>)
```

Англоязычный синоним:

```
SelectGroup
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — выбирать группы; 0 — не выбирать группы.

Возвращаемое значение:

Текущее числовое значение режима выборки групп (на момент до исполнения метода).

Описание:

Метод ВыборГруппы устанавливает режим выборки групп для формы списка справочника, которая открыта в режиме выбора или подбора элемента.

По умолчанию, выборка элементов справочников для полей в формах документов, журналов и справочников установлена без выбора групп, а в форме отчета с выбором групп. Поэтому реально имеет смысл применять данный метод только в том случае, если надо изменить режим выборки групп.

Пример:

```
// Задаем выборку без групп
ВыборГруппы(0);
```

РедактироватьВДиалоге

Установить способ редактирования элементов справочника.

Синтаксис:

```
РедактироватьВДиалоге(<Способ>, <Разрешение>)
```

Англоязычный синоним:

```
EditInForm
```

Параметры:

<Способ>	Необязательный параметр. Способ редактирования элемента справочника: 1 — в диалоге; 0 — в строке.
<Разрешение>	Необязательный параметр. Флаг разрешения пользователю менять способ редактирования: 1 — разрешить; 0 — запретить.

Возвращаемое значение:

Значение установленного на данный момент способа редактирования элементов справочника (до применения метода).

Описание:

Метод РедактироватьВДиалоге устанавливает способ редактирования элементов справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
ТекСп = РедактироватьВДиалоге();
// какой сейчас способ?
РедактироватьВДиалоге(1);
// установить редактирование в диалоге
РедактироватьВДиалоге(1, 0);
// установить редактирование в диалоге
// и запретить его менять
```

СохранениеПериодическихРеквизитов

Установить режим отображения диалога сохранения периодических реквизитов при записи элемента справочника.

Синтаксис:

```
СохранениеПериодическихРеквизитов(<ВариантВызова>, <Список>)
```

Англоязычный синоним:

```
PeriodicAttributesSaving
```

Параметры:

<ВариантВызова>	Число, определяет вариант вызова метода: 0 — отключить автоматический режим определения списка измененных реквизитов и показ диалога (параметр <Список> не используется); 1 — включить автоматический режим определения списка измененных реквизитов с показом диалога (параметр <Список> определяет набор анализируемых реквизитов); 2 — включить автоматический режим определения списка измененных реквизитов без показа диалога (параметр <Список> определяет набор анализируемых реквизитов); 3 — выполнить определение списка измененных реквизитов с показом диалога (параметр <Список> определяет набор анализируемых реквизитов); 4 — выполнить определение списка измененных реквизитов без показа диалога (параметр <Список> определяет набор анализируемых реквизитов); 5 — выдать/установить список сохраняемых периодических реквизитов. Метод возвращает установленный список периодических реквизитов, которые будут записываться на момент до вызова метода. Если параметр <Список> указан, то устанавливается список периодических реквизитов, которые должны быть записаны.
<Список>	Необязательный параметр. Строковое выражение, в котором через запятую перечислены имена периодических реквизитов справочника, которые должны выводится в диалоге (если диалог показывается), или по которым будет записан элемент истории (если диалог не показывается). По умолчанию предполагается, что в списке все периодические реквизиты справочника. Если указан символ "*", то считается, что в список включены все периодические реквизиты справочника.

Описание:

Метод СохранениеПериодическихРеквизитов управляет режимом автоматического определения списка измененных периодических реквизитов, показом диалога для выбора записываемых периодических реквизитов, а также предоставляет доступ непосредственно к списку записываемых периодических реквизитов.

Если данный метод опущен, то при записи элемента справочника диалог отображается со всеми периодическими реквизитами.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
СохранениеПериодическихРеквизитов(1, "ЦенаРозн, ЦенаОптов");
```

Сортировка

Установить способ сортировки элементов справочника.

Синтаксис:

```
Сортировка(<Способ>, <Разрешение>)
```

Англоязычный синоним:

```
SortOrder
```

Параметры:

<Способ>	Строка с именем устанавливаемой сортировки. Это может быть (в зависимости от языка по умолчанию) "Код\Code", "Наименование\Description" или имя реквизита по которому устанавливается сортировка.
<Разрешение>	Необязательный параметр. Флаг разрешения пользователю менять способ сортировки: 1 — разрешить; 0 — запретить.

Возвращаемое значение:

Возвращает текущее значение установленного на данный момент способа сортировки элементов справочника (до применения метода).

Описание:

Метод Сортировка позволяет установить способ сортировки элементов справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»).

Пример:

```
// установить сортировку по наименовнаию и запретить ее менять
Сортировка("Наименование", 0);
// получить текущее значение сортировки и установить сортировку по
// реквизиту "Оклад"
СтСорт = Сортировка("Оклад");
```

УстановитьОтбор

Установить отбор списка справочника.

Синтаксис:

```
УстановитьОтбор(<ИмяОтбора>, <Значение0тбора>)
```

Англоязычный синоним:

```
SetSelection
```

Параметры:

<ИмяОтбора>	Строковое выражение — строка с именем реквизита справочника (по которому возможен отбор — флажок свойств «Отбор по реквизиту»), по которому установлен отбор. Если это значение пустое, то отбор отключается.
<ЗначениеОтбора>	Необязательный параметр. Значение отбора.

Описание:

Метод УстановитьОтбор принудительно устанавливает отбор для списка справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
Процедура ПриОткрытии()
   Перем Тип;
   Перем Знач;
   БылОтбор = ПолучитьОтбор(Тип, Знач);
   Если .... Тогда
      // что-нибудь проверяем
      УстановаитьОтбор("Имя", "Коля");
   ИначеЕсли БылОтбор = 1 Тогда
      // вернем назад
      УстановитьОтбор(Тип, Знач);
   КонецЕсли;
КонецПроцедуры
```

ПолучитьОтбор

Возвратить текущее значение отбора списка справочника.

Синтаксис:

```
ПолучитьОтбор(<ИмяОтбора>, <Значение0тбора>)
```

Англоязычный синоним:

GetSelection

Параметры:

<ИмяОтбора>	Имя переменной, куда вернется строковое значение имени отбора.
<ЗначениеОтбора>	Имя переменной, куда вернется значение отбора.

Возвращаемое значение:

Числовое значение: 1 — есть отбор; 0 — если нет отбора.

Описание:

Метод ПолучитьОтбор возвращает текущее значение отбора списка справочника.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

См. пример для предыдущей функции.

ВидыОтбора

Установить доступные виды отборов списка справочника для вызова их в интерактивном режиме.

Синтаксис:

ВидыОтбора(<СписокИменОтборов>)

Англоязычный синоним:

KindsOfSelection

Параметры:

<СписокИменОтборов>	Строка со списком (через запятую) имен тех реквизитов, отбор по которым допускается, или символ "*" — для всех видов отборов.

Возвращаемое значение:

Строковое значение, содержащее текущий список имен отборов, разделенных запятыми (на момент до исполнения метода).

Описание:

Метод ВидыОтбора позволяет установить доступные виды отборов списка справочника для вызова их в интерактивном режиме.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
Процедура ПриОткрытии()
   ВидыОтбора("Имя, Оклад");
КонецПроцедуры
```

ЗакладкиОтбора

Установить в форме списка справочника закладки для интерактивного осуществления отбора.

Синтаксис:

ЗакладкиОтбора(<ИмяОтбора>, <Значение0тбора>)

Англоязычный синоним:

TabCtrlSelection

Параметры:

<ИмяОтбора>	Строковое выражение, содержащее имя отбора.
<ЗначениеОтбора>	Значение отбора, который будет установлен первоначально.

Описание:

Метод ЗакладкиОтбора устанавливает в форме списка справочника закладки для интерактивного осуществления отбора.

Данный метод доступен только в контексте Модуля формы списка справочника (см. «Виды программных модулей»).

Пример:

```
ЗакладкиОтбора("Имя", "Коля");
```

ИспользоватьСписокЭлементов

Реализует фильтрацию элементов справочника используя нединамический фильтр.

Синтаксис:

ИспользоватьСписокЭлементов (<СписокЗначений>)

Англоязычный синоним:

UseItemList

Параметры:

<СписокЗначений>	Необязательный параметр. Значение типа «СписокЗначений». Должен представлять собой список элементов справочника, выбранных для просмотра. Если это значение пустое, то фильтр отключается.

Описание:

Метод ИспользоватьСписокЭлементов реализует фильтрацию элементов справочника. Устанавливает нединамический фильтр.

Рекомендуется применять в предопределенных процедурах ПриСменеРодителя, ПриОткрытии, ПриСменеИерархии. При этом формируется список «разрешенных» элементов, который передается форме списка справочника при помощи данного метода. Данный список должен являться подмножеством тех элементов, которые обычно отображаются в форме списка без применения фильтра.

После вызова метода ИспользоватьСписокЭлементов форма списка справочника не позволяет вводить новый элемент, копировать и т. п.

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
Процедура УстановитьФильтр(ВыбПризнак)
   Список.СоздатьОбъект("СписокЗначений");
   Буфер = СоздатьОбъект("Справочник.Главн");
   // отбираем только те элементы, которые могут отображаться в текущем списке
   Буфер.ИспользоватьРодителя(ИспользоватьРодителя());
   Буфер.ВключатьПодчиненные(0);
   Буфер.ВыбратьЭлементы();
   Пока Буфер.ПолучитьЭлемент() = 1 Цикл
      // отбираем только те элементы, которые удовлетворяют заданному признаку
      Если (Буфер.Признак = ВыбПризнак) Тогда
         Список.ДобавитьЗначение(Буфер.ТекущийЭлемент());
      КонецЕсли;
   КонецЦикла;
   ИспользоватьСписокЭлементов(Список);
КонецПроцедуры
```

ПросмотрИстории

Установить список периодических реквизитов, просмотр истории которых допускается.

Синтаксис:

```
ПросмотрИстории(<СписокРеквизитов>)
```

Англоязычный синоним:

```
ViewHistory
```

Параметры:

<СписокРеквизитов>	Строка со списком идентификаторов (через запятую) тех реквизитов, просмотр истории которых допускается.

Возвращаемое значение:

Текущий (на момент до вызова метода) список реквизитов, просмотр истории которых допускается просмор истории.

Описание:

Метод ПросмотрИстории позволяет установить список реквизитов для которых допускается просмор истории .

Данный метод доступен в контексте Модуля формы списка справочника (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку справочника, который доступен в локальном контексте Модуля формы списка справочника.

Пример:

```
Процедура ПриОткрытии()
   ПросмотрИстории("Оклад, Тариф, Подразделение");
КонецПроцедуры
```

Предопределенные процедуры Модуля формы списка справочника

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях формы списка справочника (см. «Виды программных модулей»).

В основном данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях.

Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ПриВводеСтроки

Предопределенная процедура при вводе новой строки списка справочника.

Синтаксис:

```
ПриВводеСтроки()
```

Англоязычный синоним:

```
OnNewLine
```

Описание:

Вызов предопределенной процедуры `ПриВводеСтроки` производится в системе 1С:Предприятие при интерактивном вводе новой строки (до начала ввода) в форме списка справочника. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя вводить новые строки списка справочника), то новая строка списка справочника не будет инициирована.

Данная предопределенная процедура может располагаться в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриВводеСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права добавлять строки!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: `СтатусВозврата`

ПриРедактированииНовойСтроки

Предопределенная процедура при редактировании новой строки списка справочника.

Синтаксис:

```
ПриРедактированииНовойСтроки()
```

Англоязычный синоним:

```
OnEditNewLine
```

Описание:

Вызов предопределенной процедуры `ПриРедактированииНовойСтроки` производится в системе 1С:Пред приятие в момент начала интерактивного редактирования новой строки списка справочника (после того, как новая строка уже заведена). Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов нового элемента справочника. В данной предопределенной процедуре установка статуса возврата не

имеет смысла, т. к. отказаться от ввода новой строки в этот момент уже невозможно.

Данная предопределенная процедура может располагаться в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриРедактированииНовойСтроки()
   Оклад = 100000;
   Пдр = СоздатьОбъект("Справочник.Подразделения");
   Пдр.НайтиПоКоду(1);
   Подразделение = Пдр.ТекукщийЭлемент();
   Город = Константа.НашГород;
КонецПроцедуры
```

ПриНачалеРедактированияСтроки

Предопределенная процедура при начале редактирования существующей строки списка справочника.

Синтаксис:

```
ПриНачалеРедактированияСтроки()
```

Англоязычный синоним:

```
OnStartEditLine
```

Описание:

Вызов предопределенной процедуры `ПриНачалеРедактированияСтроки` производится в системе 1С:Пред приятие в момент начала интерактивного редактирования существующей строки списка справочника (кроме новой). Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять значения реквизитов справочника), запись не будет изменена.

Данная предопределенная процедура может располагаться в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права менять реквизиты!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура при записи строки списка справочника.

Синтаксис:

ПриЗаписи(<СписокПериодРекв>)

Англоязычный синоним:

OnWrite

Параметры:

<СписокПериодРекв>	Строковое значение — список разделенных запятыми изменяемых периодических реквизитов справочника. В данный параметр система 1С:Предприятие передает перечень периодических реквизитов, которые были интерактивно выбраны пользователем для обновления в окне диалога выбора.

Описание:

Вызов предопределенной процедуры ПриЗаписи производится системой 1С:Предприятие при интерактивной записи строки списка справочника. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять некоторые элементы справочника), то запись строки списка справочника не будет выполнена.

Формальный параметр <СписокПериодРекв> используется в теле процедуры для обработки события интерактивной записи строки списка справочника.

Данная предопределенная процедура может располагаться в следующих программных модулях: Модуль формы элемента справочника, Модуль формы группы справочника, Модуль формы списка справочника, Модуль формы списка справочника (см. Виды программных модулей).

Пример:

```
Процедура ПриЗаписи (СписокРекв)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если ТекущийЭлемент() = Константа.НашаФирма Тогда
         Предупреждение("У вас нет права менять реквизиты!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриПереносеЭлементаВДругуюГруппу

Предопределенная процедура переноса элемента справочника в другую группу.

Синтаксис:

ПриПереносеЭлементаВДругуюГруппу(<Элемент>, <Группа>)

Англоязычный синоним:

OnMoveIntoOtherGroup

Параметры:

<Элемент>	Значение элемента справочника, который переносится.
<Группа>	Значение группы справочника, в которую переносится элемент справочника.

Описание:

Вызов предопределенной процедуры ПриПереносеЭлементаВДругуюГруппу производится в системе 1С:Предприятие при интерактивном переносе элемента справочника в другую группу. Если в данной предопределенной процедуре установить статус возврата — 0, то перенос не будет произведен.

Данная предопределенная процедура может располагаться только в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриПереносеЭлементаВДругуюГруппу()
   Если НазваниеНабораПрав() = "Продавец" Тогда
```

```
      Предупреждение("У вас нет права менять группуi", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыбореРодителя

Предопределенная процедура выбора родительской группы справочника.

Синтаксис:

ПриВыбореРодителя(<Элемент>)

Англоязычный синоним:

OnSetParent

Параметры:

<Элемент>	Значение элемента справочника, который интерактивно устанавливается в качестве родителя.

Описание:

Вызов предопределенной процедуры ПриВыбореРодителя производится в системе 1С:Предприятие при интерактивной смене родительской группы справочника (выбор следующего или предыдущего уровня). Если в данной предопределенной процедуре установить статус возврата — 0, то выбор родительской группы не будет произведен.

Данная предопределенная процедура может располагаться только в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриВыбореРодителя(Родитель)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если Родитель = ЗапрещеннаяГруппа Тогда
         Предупреждение("Вам запрещено просматривать эту группу!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыбореВладельца

Предопределенная процедура выбора владельца подчиненного справочника.

Синтаксис:

ПриВыбореВладельца(<Элемент>)

Англоязычный синоним:

OnSetOwner

Параметры:

<Элемент>	Значение элемента справочника, который интерактивно устанавливается в качестве владельца подчиненного справочника.

Описание:

Вызов предопределенной процедуры ПриВыбореВладельца производится в системе ЮПредприятие при интерактивном выборе владельца подчиненного справочника (при интерактивной смене владельца, т. е. смене позиции в справочнике-владельце, которая приводит к смене отображаемых в подчиненном справочнике элементов). Если в данной предопределенной процедуре установить статус возврата — 0, то выбор владельца подчиненного справочника не будет произведен.

Данная предопределенная процедура может располагаться только в Модуле формы списка справочника (см. «Виды программных модулей»).

Пример:

```
Процедура ПриВыбореВладельца(Владелец)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если Владелец = ЗапрещенныйВладелец Тогда
         Предупреждение("Нельзя просматривать эти элементы!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриСменеИерархии

Предопределенная процедура смены режима отображения иерархии справочника.

Синтаксис:
ПриСменеИерархии(<Способ>)
Англоязычный синоним:
OnHierarchyChange
Параметры:

<Способ>	Значение устанавливаемого (тот, который пользователь хочет установить) способа просмотра справочника: 1 — иерархических список; 0 — все элементы сразу.

Описание:
Вызов предопределенной процедуры ПриСменеИерархии производится в системе 1С:Предприятие при интерактивной смене режима отображения иерархии справочника (пункт меню «Иерархический список»). Если в данной предопределенной процедуре установить статус возврата — 0, то смена режима отображения иерархии справочника не будет произведена.
Данная предопределенная процедура может располагаться только в Модуле формы списка справочника (см. «Виды программных модулей»).
Пример:

```
Процедура ПриСменеИерархии(ВыбСпособ)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("Нельзя менять режим просмотра 1", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУстановкеОтбора

Предопределенная процедура при установке отбора справочника.
Синтаксис:
ПриУстановкеОтбора(<ТипОтбора>, <Значение0тбора>)
Англоязычный синоним:
OnSetSelectInJournal
Параметры:

<ТипОтбора>	Строковое значение — тип устанавливаемого отбора (имя реквизита справочника по которому устанавливается отбор).
<Значение0тбора>	Устанавливаемое значение отбора.

Описание:
Вызов предопределенной процедуры ПриУстановкеОтбора производится в системе 1С:Предприятие при интерактивной установке отбора любым способом (отбор, быстрый отбор, отбор по значению, история отбора) и при отключении отбора. Если в данной предопределенной процедуре установить статус возврата — 0, то установка отбора справочника не будет произведена.
Данная предопределенная процедура может располагаться только в Модуле формы списка справочника (см. «Виды программных модулей»).
Пример:

```
Процедура ПриУстановкеОтбора(ВыбСпособ)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("Нельзя устанавливать отбор;", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 11
Работа с Перечислениями

Перечисление — средство работы с элементами данных, список возможных значений которых жестко задан в конфигурации. Например, для перечисления "ТипСотрудника" можно задать возможные значения: "Штатный", "Совместитель".

В отличие от справочника, списки значений в перечислении задаются исключительно в процессе их создания в конфигураторе и при выполнении задачи не могут быть изменены.

Контекст работы с перечислениями

Средства языка предоставляют возможность непосредственного доступа к заданным в конфигураторе значениям перечислений в любом программном модуле (перечисления принадлежат глобальному контексту задачи). В терминах языка перечисления аналогичны системным константам «только для чтения», т. е. идентификаторы перечислений могут размещаться только в правой части оператора присваивания, в выражениях, быть параметрами процедур, функций или методов в любом программном модуле.

В качестве имени перечисления обязательно должно выступать полное имя конкретного вида и значения перечисления, как оно объявлено в конфигураторе. Вид и значение перечисления записывается через точку после ключевого слова Перечисление, т. е. полное имя вида перечисления записывается следующим образом:

Перечисление.<Имя_Переч>.<Значение_Переч>,

где <Имя_Переч> — имя вида перечисления, <Значение_Переч> — конкретное значение данного вида перечисления.

Англоязычный синоним ключевого слова Перечисление — Enum.

Пример:

```
Если Док.Сотрудник.Тип = Перечисление.ТипСотрудника.Штатный Тогда
   Льготы = 1;
Иначе
   Льготы = 0;
КонецЕсли;
```

Объект Перечисление является частью глобального контекста программы 1С:Предприятие. Этот объект в качестве своих атрибутов имеет значения объектов конкретных видов перечислений, заданных в конфигурации. Кроме того, этот объект имеет метод ПолучитьАтрибут, который позволяет получить доступ к объекту перечисления конкретного вида по его имени.

Перечисление конкретного вида в качестве своих атрибутов имеет конкрет-йьте значения перечислений. Кроме того, этот объект имеет методы КоличествоЗначений, ЗначениеПоНомеру и ЗначениеПоИдентификатору которые позволяют получить доступ к конкретному значению перечисления.

Конкретное значение перечисления имеет методы Вид, ПредставлениеВида, Выбран, ПорядковыйНомер, Идентификатор.

Методы перечислений

ПолучитьАтрибут

Получить доступ к объекту перечисления конкретного вида по его имени.

Синтаксис:

ПолучитьАтрибут(<ИмяПеречисления>)

Англоязычный синоним:

GetAttrib

Параметры:

<ИмяПеречисления>	Строковое выражение, содержащее имя вида перечисления, как оно задано в конфигураторе.

Возвращаемое значение:

Объект перечисления конкретного вида.

Описание:

Метод ПолучитьАтрибут позволяет получить доступ к объекту перечисления конкретного вида по его имени, как оно задано в конфигураторе.

Этот метод применяется только к объекту глобального контекста Перечисление.

Пример:

```
Булево = Перечисление.ПолучитьАтрибут("Булево");
```

КоличествоЗначений

Определить общее количество значений в данном виде перечисления.

Синтаксис:

```
КоличествоЗначений()
```

Англоязычный синоним:

```
Count
```

Возвращаемое значение:

Число общего количества значений в данном виде перечисления.

Описание:

Метод КоличествоЗначений применяется к конкретному виду перечисления и позволяет определить общее количество значений в данном виде перечисления.

Пример:

```
// отобразим все значения перечисления
Всего = Перечисление.ВидыТоваров.КоличествоЗначений();
Пока Ном = 1 По Всего Цикл
   Сообщить(Перечисление.ВидыТоваров.ЗначениеПоНомеру(Ном));
КонецЦикла;
```

ЗначениеПоНомеру

Определить значение перечисления, соответствующее номеру позиции в конфигураторе.

Синтаксис:

```
ЗначениеПоНомеру(<Номер>)
```

Англоязычный синоним:

```
ValueByIndex
```

Параметры:

<Номер>	Числовое выражение. Номер позиции значения перечисления, заданный в конфигураторе.

Возвращаемое значение:

Значение перечисления, соответствующее номеру заданной позиции.

Описание:

Метод ЗначениеПоНомеру применяется к конкретному виду перечисления и позволяет определить значение перечисления, соответствующее номеру позиции в конфигураторе.

Пример:

См. предыдущий пример.

ЗначениеПоИдентификатору

Определить значение перечисления, соответствующее идентификатору в конфигураторе.

Синтаксис:

```
ЗначениеПоИдентификатору(<Идентификатор>)
```

Англоязычный синоним:

```
ValueByIdentifier
```

Параметры:

<Идентификатор>	Строковое выражение. Идентификатор перечисления, заданный в конфигураторе.

Возвращаемое значение:

Значение перечисления, соответствующее идентификатору в конфигураторе. Если не найдено — то пустое значение.

Описание:

Метод ЗначениеПоИдентификатору применяется к конкретному виду перечисления и позволяет определить значение перечисления, соответствующее идентификатору в конфигураторе.

Пример:

См. предыдущий пример.

Вид

Определить название вида перечисления.

Синтаксис:

```
Вид()
```

Англоязычный синоним:

```
Kind
```

Возвращаемое значение:

Строковое значение, содержащее название вида перечисления.

Описание:

Метод Вид позволяет определить название вида перечисления, как оно задано в конфигураторе.

Пример:
```
// отобразим вид перечисления в строке состояния
Состояние(ИзмерениеТовара.Вид());
```

ПредставлениеВида

Определить пользовательское представление вида перечисления.

Синтаксис:
```
ПредставлениеВида()
```
Англоязычный синоним:
```
KindPresent
```
Возвращаемое значение:

Строковое значение, содержащее пользовательское представление вида перечисления (синоним перечисления или, если он пустой, то идентификатор).

Описание:

Метод `ПредставлениеВида` позволяет получить пользовательское представление вида перечисления, как оно задано в конфигураторе.

Пример:
```
// отобразим пользовательское представление в строке состояния
Состояние(ИзмерениеТовара.ПредставлениеВида());
```

Выбран

Возвратить флаг выбора элемента перечисления.

Синтаксис:
```
Выбран()
```
Англоязычный синоним:
```
Selected
```
Возвращаемое значение:

Числовое значение: 1 — если элемент перечисления выбран (спозиционирован); 0 — если не выбран.

Описание:

Метод `Выбран` возвращает число со значением 1 — если элемент перечисления выбран (спозиционирован), 0 — если элемент перечисления не выбран.

Пример:
```
// В диалоге формирования некоторого отчета
// ВыбЕдИзмер – реквизит диалога типа «Перечисление.Единицы»
Если ВыбЕдИзмер.Выбран() = 0 Тогда
   // Если единица измерений в диалоге не выбрана, то формируем отчет без условий
   Заг1 = "По всем единицам измерения.";
Иначе
   // если в диалоге выбрана единица измерений
   // то формируем отчет только по выбранным единицам
   Заг1 = "Отчет по товарам с единицей измерения " + ВыбЕдИзмер;
КонецЕсли;
```

ПорядковыйНомер

Определить порядковый номер значения в перечеслении.

Синтаксис:
```
ПорядковыйНомер()
```
Англоязычный синоним:
```
Number
```
Возвращаемое значение:

Строковое значение, содержащее название вида перечисления.

Описание:

Метод `ПорядковыйНомер` определяет порядковый номер значения в перечеслении, как он задан в конфигураторе. Значения перечисления нумеруются с 1.

Пример:
```
// отобразим порядковый номер значения перечисления
Сообщить(Перечисление.Единицы.Штука.ПорядковыйНомер());
```

Идентификатор

Возвращает строку — идентификатор значения перечисления.

Синтаксис:

Идентификатор()

Англоязычный синоним:

Identifier

Возвращаемое значение:

Строковое значение — идентификатор значения перечисления как он задан в метаданных.

Описание:

Метод Идентификатор возвращает строку — идентификатор значения перечисления как он задан в метаданных.

Пример:

```
// отобразим все идентификаторы перечисления
Всего = Перечисление.ВидыТоваров.КоличествоЗначений();
Пока Ном = 1 По Всего Цикл
   ВидТов = Перечисление.ВидыТоваров.ЗначениеПоНомеру(Ном);
   Сообщить(ВидТов.Идентификатор());
КонецЦикла;
```

Глава 12
Работа с Документами

Документ — одно из основных понятий системы 1С:Предприятие. При помощи документов организуется ввод в систему информации о совершаемых хозяйственных операциях, а также ее просмотр и, если необходимо, корректировка.

В большинстве своем документы, которые создаются в процессе настройки конфигурации задачи, являются электронными аналогами стандартных бумажных документов, являющихся основаниями для тех или иных учетных действий или расчетов. Тем на менее, использование этого типа данных может выходить далеко за рамки простой фиксации изменений внесенных в регистры и журналы расчетов.

Структура каждого конкретного вида документа определяется при его создании в конфигураторе. У любого вида документа существует два обязательных реквизита, которые создаются автоматически — дата и номер документа. Другие реквизиты документа определяются в конфигураторе конкретно для каждого создаваемого вида документа.

В документах выделяются две основные структурные части: шапка документа и многострочная табличная часть, поэтому реквизиты документа можно подразделить на «Реквизиты шапки» и «Реквизиты табличной части».

Как правило, в шапке содержатся реквизиты, которые являются общими для всего документа. Реквизиты шапки принадлежат всему документу в целом и принимают только одно значение.

Например, в случае расчета заработной платы, документ «Больничный лист» в заголовочной части может содержать процент оплаты, сведения о сотруднике.

Многострочная (табличная) часть документа представляет собой список однотипных строк. Реквизиты табличной части принадлежат строке документа, т. е. каждая строка содержит свои собственные значения этих реквизитов.

Например, в уже упоминавшемся больничном листе табличная часть — это набор данных за прошлые расчетные периоды, предназначенные для расчета средней заработной платы.

Контекст работы с документами

В синтаксисе языка доступ к атрибутам, а также вызов методов документов зависит от контекста выполнения программного модуля.

Если конкретный документ входит (согласно локального контекста) в набор непосредственно доступных модулю значений агрегатных типов данных (см. «Виды программных модулей»), то доступ к атрибуту, вызов метода для этого документа — просто имя этого атрибута, метода с указанием необходимых параметров.

Пример:

* В форме редактирования документа «Накладная» мы имеем непосредственный доступ к текущему документу (накладной). Значит, чтобы изменить номер накладной, запишем:

```
НомерДок = "12345678";
```

* В форме редактирования документа «Приказ на зачисление» мы имеем непосредственный доступ к реквизитам этого документа. Значит, чтобы записать имя нового сотрудника, пишем:

```
ФИО = "Иванов И.И.";
```

Значение документа может быть получено из других источников, например как реквизит другого документа. В этом случае обращение к атрибутам и методам такого документа представляет собой сложное выражение, где имена реквизитов разделяются точкой.

Пример:

* Допустим, у документа «Счет» есть реквизит «Накладная», который имеет тип «Документ.РасхНакл». Номер накладной можно получить следующим образом:

```
НомерРасхНакл=Документ.Счет.Накладная.НомерДок;
```

* Допустим, значение реквизита «Приказ» справочника «ПриказыНаДоплату» имеет тип «Документ.ПриказНаДоплату». Тогда номер приказа можно получить следующим образом:

```
Спр = СоздатьОбъект("Справочник.ПриказыНаДоплату");
// ...
Номер = Спр.Приказ.НомерДок;
```

В других случаях, доступ к атрибутам, вызов методов конкретного документа происходит при помощи переменной со ссылкой на объект типа «Документ». Объект создается функцией `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать атрибут или метод объекта, имя этого атрибута, метода (с указанием необходимых параметров) пишется через точку после имени ссылки.

При создании ссылки на объект типа «Документ» при помощи функции `СоздатьОбъект` в качестве имени документа может выступать ключевое слово `Документ` или полное имя вида документа.

Полное имя вида документа записывается следующим образом:

`Документ.<Имя_Документа>`

где <Имя_Документа> — имя вида документа, как он объявлен в конфигураторе.

Применение ключевого слова "Документ" (без указания вида документа) используется для организации доступа ко всему перечню документов независимо от вида. В этом случае методы `ВыбратьДокументы` и `ПолучитьДокумент` будут обрабатывать документы всех видов. Однако, для переменных, созданных таким вызовом функции `СоздатьОбъ-`

ект, будут недоступны процедуры поиска, выбора документа из диалога, ввода нового документа, которые подразумевают конкретный вид документа.

Англоязычный синоним ключевого слова Документ — Document.

Замечание: Следует обратить особое внимание, что переменная типа «Документ», созданная функцией СоздатьОбъект — это ссылка на список документов в отличие от переменных содержащих само значение объекта (например, переменной может быть присвоено значение реквизита справочника, который имеет тип «Документ»). Использование ссылки на список документов, созданной при помощи функции СоздатьОбъект существенно отличается от работы со значением типа «Документ». Только при работе со ссылкой на список документов разрешено изменять позицию (найти-выбрать...) текущего элемента в списке (т. е. осуществлять позиционирование по списку документов), создавать новые, изменять и удалять существующие документы. С другой стороны, ссылка на список документов не содержит собственно значения конкретного документа, которое можно присвоить чему-либо. Однако, его всегда можно получить, используя функцию ТекущийДокумент.

Замечание. Объект, созданный при помощи функции СоздатьОбъект, изначально не определен, т. е. не содержит никакого значения. Чтобы начать с ним работать, его предварительно надо позиционировать (установить на конкретный документ) при помощи процедур НайтиДокумент, НайтиПоНомеру, ПолучитьДокумент и т. п..

Пример:

```
*
Док = СоздатьОбъект("Документ");
Док1 = СоздатьОбъект("Документ.БольничныйЛист");
Док2 = СоздатьОбъект("Документ.ПриказНаДоплату");
Док3 = СоздатьОбъект("Документ.ПриказПоОтпуску");
Док4 = СоздатьОбъект("Документ.ПрихНакладн");
Док5 = СоздатьОбъект("Документ.Списание");
*
// В модуле формы документа
// меняем номер обрабатываемого документа
НомерДок = "12345";
*
// В других модулях имена атрибутов, процедур и функций документов
// записываются через точку после имени переменной
// со значением типа "Документ".
Док = СоздатьОбъект("Документ.ПриказыДоплат");
// создаем новый документ
Док.Новый();
Док.ОбщийРазмер = 12000000;
// меняем номер Документа
Док.НомерДок = "12345";
// меняем дату документа
Док.ДатаДок = '14.04.96';
// создаем новую строку в спецификации документа
Док.НоваяСтрока();
Док.Сотр = СпрСотрудн.ТекущийЭлемент();
Док.Процент = 15;
// ...
// записываем документ
Док.Записать();
```

Позиция документа

Все документы в системе 1С:Предприятие располагаются на временной оси. Основными характеристиками расположения документа на временной оси являются дата и время документа. Дата документа доступна через атрибут ДатаДок, а работа со временем документа осуществляется при помощи методов УстановитьВремя, ПолучитьВремя и предопределенной процедуры ПриИзмененииВремениДокумента. Кроме даты и времени в системе 1С:Предприятие существует понятие позиции документа.

Позиции документа — это строковое значение специального формата длиной 32 символа. Позиция документа однозначно характеризует расположение документа на оси времени относительно других документов, т. е. позиция документа заключает в себе и дату и время (с точностью до секунды), плюс некоторые дополнительные данные, определяющие вза-

имное расположение документов в пределах одной секунды. Позиции документов можно сравнивать (>; <), определяя какой документ раньше, какой позже. Позиции документов можно сортировать по возрастанию или по убыванию. Кроме того, можно получить позицию точки актуальности (ПолучитьПозициюТА) и позицию границы последовательности (ПолучитьПозицию). Позицию документа можно получить при помощи метода Позиция.

Многие методы могут в качестве парамера воспринимать кроме документа или даты позицию документа.

См. также:

СформироватьПозициюДокумента, РазобратьПозициюДОкумента, ПолучитьПозициюТА, ПолучитьПозицию, ВыбратьДокументы, ВыбратьПодчиненныеДокументы, ВыбратьПоЗначению, РассчитатьРегистрыНа, РассчитатьРегистрыПо, УстановитьТАна, УстановитьТАпо, ВыбратьДвижения, ВыбратьДвиженияСОстатками, ВыполнитьЗапрос, Рассчитать, ВыбратьОперации, ВыбратьОперациисПроводками, ВыбратьПоЗначению

Атрибуты документов

НомерДок

Номер документа

Синтаксис:

НомерДок

Англоязычный синоним:

DocNum

Описание:

При помощи атрибута НомерДок можно получать и задавать значение номера выбранного документа.

Пример:

```
Основание = Вид() + " № " + СокрП(НомерДок) + " от " + ДатаДок;
```

ДатаДок

Дата документа

Синтаксис:

ДатаДок

Англоязычный синоним:

DocDate

Описание:

Атрибут ДатаДок задает значение даты выбранного документа.

Пример:

```
Процедура ВводНового()
   // устанавливаем дату документа
   ДатаДок=РабочаяДата();
   Подразделение = Константа.АУП;
   Валюта = Константа.ДефВалютаЗакупки;
   Валюта.ИспользоватьДату(ДатаДок);
   Дата_Курса = ДатаДок;
   Курс = Валюта.Текущ_курс;
   Фирма = Константа.ДефФирма;
КонецПроцедуры
```

<Реквизит>

Значение реквизита документа.

Синтаксис:

<Реквизит> Идентификатор реквизита документа, как он задан в конфигураторе.

Описание:

Атрибут <Реквизит> задает значение реквизита текущего документа. В тексте программного модуля используется идентификатор конкретного реквизита документа, созданного в конфигураторе. Реквизиты многострочной части документа имеют смысл только при выбранной строке многострочной части документа.

Пример:

```
// предопределенная процедура ввода нового документа
// ТипСотр, Оклад – реквизиты документа
Процедура ВводНового()
   // Установим дату документа
   ДатаДок = РабочаяДата();
   ТипСотр = Перечисление.ТипСотрудника.Штатный;
```

```
   Оклад = Сотр.Оклад.Получить(ДатаДок);
   ПризнакНакладной = Перечисление.ПризнРасхНакл.Продажа;
КонецПроцедуры
```

НомерСтроки

Синтаксис:
НомерСтроки
Англоязычный синоним:
LineNum
Описание:
Атрибут НомерСтроки задает числовое значение номера строки многострочной части выбранного документа. Данный атрибут имеет смысл только при выбранной строке. Присвоение строке нового номера передвигает строку в многострочной части документа.
Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

Операция

Атрибут «Операция» предназначен для обращения к бухгалтерской операции документа.
Синтаксис:
Операция
Англоязычный синоним:
Operation
Описание:
Атрибут «Операция» документа используется только в случае, если установлена компонента «Бухгалтерский учет» и только для тех видов документов, для которых в конфигурации установлен признак «Бухгалтерский учет». Он предназначен для обращения к операции документа. Данный атрибут не используется как самостоятельное значение, а позволяет обращаться к атрибутам и методам операции. Описание атрибутов и методов операции см. в разделе «Работа с операциями и проводками».
Пример:
* В форме документа устанавливается содержание операции.

```
Процедура ПриЗаписи()
   Операция.Содержание = "Износ за " + Формат(ДатаДок, "Д ММММГГТГ");
КонецПроцедуры
```

Методы документов

Вид

Название вида документа.
Синтаксис:
Вид()
Англоязычный синоним:
Kind
Возвращаемое значение:
Строковое значение, содержащее название вида документа, как он задан в конфи гураторе.
Описание:
Метод Вид возвращает название вида документа, как он задан в конфигураторе.
Пример:

```
Если (Док.Вид() = "ДоплатаПроцентом") ИЛИ (Док.Вид() = "ДоплатаСуммой") Тогда
   Сотр = Док.Сотрудник;
Иначе
```

```
    Предупреждение("Это не доплата!");
КонецЕсли;
```

ПредставлениеВида

Определить пользовательское представление вида документа.

Синтаксис:

```
ПредставлениеВида()
```

Англоязычный синоним:

```
KindPresent
```

Возвращаемое значение:

Строковое значение, содержащее пользовательское представление вида документа (синоним документа или, если он пустой, то идентификатор).

Описание:

Метод `ПредставлениеВида` позволяет получить пользовательское представление вида документа, как оно задано в конфигураторе.

Пример:

```
// отобразим пользовательское представление в строке состояния
Состояние(ВыбДокум.ПредставлениеВида());
```

УстановитьАтрибут

Установить значение реквизита по имени идентификатора.

Синтаксис:

```
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SetAttrib
```

Параметры:

`<ИмяРеквизита>`	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.
`<Значение>`	Выражение, содержащее устанавливаемое значение реквизита.

Описание:

Метод `УстановитьАтрибут` позволяет установить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
Док.УстановитьАтрибут("ЦенаРозн", ЦенаТов);
```

ПолучитьАтрибут

Получить значение реквизита но имени идентификатора.

Синтаксис:

```
ПолучитьАтрибут(<ИмяРеквизита>)
```

Англоязычный синоним:

```
GetAttrib
```

Параметры:

`<ИмяРеквизита>`	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.

Возвращаемое значение:

Значение реквизита `<ИмяРеквизита>`.

Описание:

Метод `ПолучитьАтрибут` позволяет получить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
ЦенаДок = Док.ПолучитьАтрибут("Цена");
```

Выбран

Возвратить флаг выбора документа.

Синтаксис:

```
Выбран()
```

Англоязычный синоним:

```
Selected
```

Возвращаемое значение:

Числовое значение: 1 — если документ выбран, 0 — если документ не выбран.

Описание:

Метод `Выбран` позволяет проверить сам факт позиционирования объекта типа «документ».

Пример:
```
Док = СоздатьОбъект("Документ.ПрихНакл");
// позиционируем документ по номеру
Док.НайтиПоНомеру("1", 0);
Если Док.Выбран() = 1 Тогда
    // если документ найден
    // открываем выборку строк спецификации документа
    Док.ВыбратьСтроки();
    // цикл выбора строк спецификации документа
    Если Док.КоличествоСтрок() > 0 Тогда
       Пока Док.ПолучитьСтроку() = 1 Цикл
          Док.УдалитьСтроку();
       КонецЦикла;
    КонецЕсли;
КонецЕсли;
```

Проведен

Возвратить флаг проводки документа.

Синтаксис:
```
Проведен()
```
Англоязычный синоним:
```
IsTransacted
```
Возвращаемое значение:
Числовое значение: 1 — если документ проведен, 0 — если документ не проведен.

Описание:
Метод `Проведен` позволяет проверить сам факт проведения документа.

Пример:
```
Процедура ПолучитьПроведенные()
   Док = СоздатьОбъект("Документ");
   // Откроем выборку документов
   Док.ВыбратьДокументы(ДатаНачала, ДатаКонца);
   // Цикл выбора документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      Если Док.Проведен() > 0 Тогда
         // для проведенных документов выведем сообщение
         Сообщить(Строка(Док.Вид())+" № "+Док.НомерДок+" от "+Док.ДатаДок);
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

ТекущийДокумент

Определить значение текущего документа.

Синтаксис:
```
ТекущийДокумент()
```
Англоязычный синоним:
```
CurrentDocument
```
Возвращаемое значение:
Значение текущего документа.

Описание:

Метод `ТекущийДокумент` возвращает значение позиционированного текущего документа (в целом, как объекта). Данный метод применяется, например, если нужно документ передать как параметр в вызове какого-либо метода или присвоить какому-либо реквизиту.

Пример:
```
Если Режим = "Подробный" Тогда
   // используем объект типа «документ» неопределенного вида
   Док = СоздатьОбъект("Документ");
   // Откроем выборку документов
   Док.ВыбратьДокументы(ДатаНачала, ДатаОкончания);
   // Цикл выбора документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      // присвоим переменной ТекДок значение найденного документа
```

```
      ТекДок = Док.ТекущийДокумент();
      // ...
   КонецЦикла;
КонецЕсли;
```

Выбрать

Вызов диалога для выбора документа.

Синтаксис:

```
Выбрать(<Подсказка>, <ФормаЖурнала>, <КомуПодч>)
```

Англоязычный синоним:

```
Choose
```

Параметры:

<Подсказка>	Текст заголовка окна диалога ввода. Может использоваться в качестве подсказки конечному пользователю.
<ФормаЖурнала>	Строковое выражение идентификатора той формы журнала документа (как он объявлен в конфигураторе), которая должна использоваться для выбора. Если это значение пустое, то используется форма журнала по умолчанию.. Можно указывать имя объекта в следующем виде: • "Журнал.XXXXX", где XXXXX — имя вида соответствующего объекта, как он задан в конфигураторе, например: "Справочник.Товары"; • "Журнал.Подчиненные"; • "ЖурналОпераций"; • "ЖурналОпераций.YYYYY", где YYYYY — имя формы журнала операций, как оно задано в конфигураторе;
<КомуПодч>	Необязательный параметр. Используется при открытии выбора по журналу подчиненных документов, т.е когда второй параметр <ФормаЖурнала> имеет значение "Журнал.Подчиненные". В данном случае в этом параметре передается документ-владелец, по которому следует построить журнал подчиненных документов.

Возвращаемое значение:

Числовое значение: 1 — если документ выбран; 0 — если не выбран.

Описание:

Метод Выбрать вызывает диалоговое окно для выбора документа и затем позиционирует выбранный документ в качестве текущего. Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Док = СоздатьОбъект("Документ.НаклПрих");
// вызываем диалог выбора документа
Если Док.Выбрать("Найди документ") > 0 Тогда
   // удаляем выбранный документ
   Док.Удалить();
КонецЕсли;
```

См. также: ВидыДляВыбора

ВидыДляВыбора

Установка выбираемых видов для объекта типа «документ» неопределенного вида.

Синтаксис:

```
ВидыДляВыбора(<СписокВидов>)
```

Англоязычный синоним:

```
KindsForChoice
```

Параметры:

<СписокВидов>	Необязательный параметр. Строковое выражение содержащее список видов выбираемых документов, разделенных запятыми.

Возвращаемое значение:

Строковое значение, содержащее текущий список видов выбираемых документов, разделенных запятыми (на момент до исполнения метода).

Описание:

Метод ВидыДляВыбора устанавливает выбираемые виды для объекта-документ неопределенного вида. Данный метод обычно используется до начала интерактивного позиционирования документа, например, при помощи метода Выбрать.

Данный метод может использоваться только для объектов типа «документ» неопределенного вида — либо созданных функцией СоздатьОбъект, либо определенных в конфигураторе как реквизиты диалога или другого объекта. Если ме-

тод ВидыДляВыбора применен к реквизиту диалога типа «документ» неопределенного вида, то интерактивный выбор будет производиться только среди установленных видов документов.

Пример:

```
// данная процедура присваивает некоторому реквизиту "НаОсновании"
// значение конкретного документа
Процедура УстДокОснования()
   // Создадим объект типа «документ» неопределенного вида
   Дкм = СоздатьОбъект("Документ");
   Дкм.ВидыДляВыбора("РасходнаяНакл, Счет");
   // Вызываем диалог выбора документа
   Если Дкм.Выбрать("Выберите документ основания", "") > 0 Тогда
      НаОсновании = Дкм.ТекущийЭлемент();
   КонецЕсли;
КонецПроцедуры
```

См. также: Выбрать

Итог

Итоговое значение реквизита документа.

Синтаксис:

Итог(<ИмяРеквизита>)

Англоязычный синоним:

Total

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита табличной части документа, для которого в конфигураторе установлено свойство «Итог по колонке».

Возвращаемое значение:

Числовое значение — сумма по всем строкам табличной части документа для реквизита <ИмяРеквизита>.

Описание:

Метод Итог позволяет определить сумму значений по всем строкам табличной части документа для реквизита <ИмяРеквизита>.

Данный метод может быть использован только для реквизитов табличной части документов, для которых установлено свойство «Итог по колонке» (закладка «Дополнительные» свойств реквизита документа в конфигураторе).

Пример:

```
Процедура ВычСреднего()
   // вычислим среднюю сумму по наряду
   Сумма = Итог("Сумма");
   Штук = Итог("Штук");
   Если Штук <> 0 Тогда
      Средн = Сумма / Штук;
   Иначе;
      Средн = 0;
   КонецЕсли;
КонецПроцедуры
```

КоличествоСтрок

Определить количество строк в документе.

Синтаксис:

КоличествоСтрок()

Англоязычный синоним:

LinesCnt

Возвращаемое значение:

Числовое значение — количество строк в документе.

Описание:

Метод КоличествоСтрок позволяет определить количество строк в многострочной части документа.

Пример:

```
Сообщить("Документ: " + Вид() + " № " + НомерДок + " от " + ДатаДок +
                    " в документе " + КоличествоСтрок() + " строк");
```

НайтиДокумент

Найти документ по значению.

Синтаксис:

НайтиДокумент(<Документ>)

Англоязычный синоним:

FindDocument

Параметры:

<Документ>	Выражение, содержащее значение типа «Документ».

Возвращаемое значение:

Число 1 — если действие выполнено (документ найден);
Число 0 — если действие не выполнено.

Описание:

Метод НайтиДокумент выполняет поиск документа по значению, заданному параметром <Документ>. Данный метод используется для позиционирования объекта на конкретный документ.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Процедура УдалДок(Докум)
   Док = СоздатьОбъект("Документ.НаклПрих");
   // позиционируем созданный объект на заданном документе
   Док.НайтиДокумент(Докум);
   Если Док.Выбран() > 0 Тогда
      // если документ найден, то удалим его
      Док.Удалить();
   Иначе
      Предупреждение ("Неверно задан документ!");
   КонецЕсли;
КонецПроцедуры
```

НайтиПоНомеру

Найти документ по номеру.

Синтаксис:

НайтиПоНомеру(<Номер>, <Дата>, <ИдентВида>)

Англоязычный синоним:

FindByNum

Параметры:

<Номер>	Строковое выражение, содержащее значение номера искомого документа.
<Дата>	Выражение типа «дата».
<ИдентВида>	Необязательный параметр. Строковое выражение, содержащее идентификатор вида документа или идентификатор Нумератора.

Возвращаемое значение:

Число 1 — если действие выполнено (документ найден);
Число 0 — если действие не выполнено.

Описание:

Метод НайтиПоНомеру позиционирует документ по номеру. В качестве второго параметра задается любая дата из диапазона, в котором нужно искать документ с данным номером. Поиск зависит от выбранного в конфигураторе способа уникальности номеров (по месяцу, году и др.).

Метод может быть использован для объекта Документ общего вида, тогда для поиска нужно указать в параметре <ИдентВида> идентификатор вида документа или идентификатор Нумератора.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Если ВвестиЧисло(Ном, "Введите номер приказа", 5, 0) = 1 Тогда
   Док=СоздатьОбъект("Документ.ПриказыУвольнения");
   // позиционируем документ по заданному номеру
   Док.НайтиПоНомеру(Строка(Ном), Дата(0));
   Если Док.Выбран() = 1 Тогда
      Документ=Док.ТекущийДокумент();
      // ...
   КонецЕсли;
КонецЕсли;
```

Получить Строку ПоНомеру

Получить строку документа по номеру.

Синтаксис:

ПолучитьСтрокуПоНомеру(<Номер>)

Англоязычный синоним:
GetLineByNum
Параметры:

<Номер>	Выражение, содержащее номер искомой строки.

Возвращаемое значение:
Число 1 — если действие выполнено;
Число 0 — если действие не выполнено.
Описание:
Метод ПолучитьСтрокуПоНомеру устанавливает в качестве текущей строки спецификации документа строку с заданным номером (не порядковым, а тем, который записан в строке, т. к. могут быть пропуски).
Пример:

```
Процедура УдалЗаданнойСтроки(Док, Стр)
   Если Док.Выбран() = 1 Тогда
      // если переданный параметр содержит значение документа, то
      // найдем в нем строку
      Док.ПолучитьСтрокуПоНомеру(Стр);
      // удалим найденную строку спецификации документа
      Док.УдалитьСтроку();
   Иначе
      Предупреждение("Документ не выбран!");
   КонецЕсли;
КонецЕсли;
```

ВыбратьДокументы

Открыть выборку документов.
Синтаксис:
ВыбратьДокументы(<НачалоВыборки>, <КонецВыборки>)
Англоязычный синоним:
SelectDocuments
Параметры:

<НачалоВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки документов. Если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа.
<КонецВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки документов. Если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом.

Возвращаемое значение:
Число 1 — если действие выполнено и в выборке есть хотя бы один документ;
Число 0 — если действие не выполнено или в выборке нет ни одного документа.
Описание:
Метод ВыбратьДокументы открывает выборку документов в интервале с <НачалоВыборки> по <КонецВыборки>.
Непосредственно сама выборка осуществляется при помощи метода ПолучитьДокумент в порядке возрастания даты и времени записи документов (если не задан обратный порядок методом ОбратныйПорядок).
Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.
Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект("Документ");
   // открываем выборку документов
   Док.ВыбратьДокументы(ДатаНачало, ДатаКонец);
   // цикл получения документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      ТекДок = Док.ТекущийДокумент();
   КонецЦикла;
КонецЕсли;
```

См. также: ПолучитьДокумент, ОбратныйПорядок

ВыбратьПодчиненныеДокументы

Открыть выборку подчиненных документов.
Синтаксис:
ВыбратьПодчиненныеДокументы(<НачалоВыборки>, <КонецВыборки>, <Докум>)

Англоязычный синоним:
SelectChildDocs
Параметры:

<НачалоВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки документов. Если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа.
<КонецВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки документов. Если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом.
<Докум>	Выражение типа «документ», содержащее значение документа-владельца, подчиненные которому будут включаться в выборку.

Возвращаемое значение:
Число: 1 — если действие выполнено и в выборке есть хотя бы один документ; 0 — если действие не выполнено или в выборке нет ни одного документа.
Описание:
Метод ВыбратьПодчиненныеДокументы открывает выборку всех документов, подчиненных заданному документу <Докум> в интервале с <НачалоВыборки> по <КонецВыборки>. Непосредственно сама выборка осуществляется при помощи метода ПолучитьДокумент в порядке возрастания даты и времени записи документов (если не задан обратный порядок методом ОбратныйПорядок).
Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.
Пример:

```
//Проверка того, что по Счету были документы отгрузки
ДокТест = СоздатьОбъект("Документ");
// открываем выборку документов
ДокТест.ВыбратьПодчиненныеДокументы('01.01.80', '31.12.99', ПоСчету);
// получение документа
Если ДокТест.ПолучитьДокумент() = 1 Тогда
   Предупреждение("По данному Счету были отгрузки!");
КонецЕсли;
```

См. также: ПолучитьДокумент, ОбратныйПорядок

ВыбратьПоЗначению

Открыть выборку документов по значению.
Синтаксис:
ВыбратьПоЗначению(<НачалоВыборки>, <КонецВыборки>, <ИмяОтбора>, <Знач>)
Англоязычный синоним:
SelectByValue
Параметры:

<НачалоВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки документов. Если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа.
<КонецВыборки>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки документов. Если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом.
<ИмяОтбора>	Строковое выражение, которое содержит либо название «общего реквизита» документов, либо название «графы отбора» журналов, как они заданы в конфигураторе.
<Знач>	Значение отбора, по которому строится выборка документов.

Возвращаемое значение:
Число 1 — если действие выполнено и в выборке есть хотя бы один документ; 0 — если действие не выполнено или в выборке нет ни одного документа.
Описание:
Метод ВыбратьПоЗначению открывает выборку документов в интервале с <НачалоВыборки> по <КонецВыборки>, для которых реквизит отбора <ИмяОтбора> имеет конкретное заданное значение <Знач>.
Непосредственно сама выборка осуществляется при помощи метода ПолучитьДокумент в порядке возрастания даты и времени записи документов (если не задан обратный порядок методом ОбратныйПорядок).
Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.
Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект("Документ");
   // открываем выборку документов
   Док.ВыбратьПоЗначению(ДатаНачало, ДатаКонец, "Автор", "Сидоров");
```

```
   // цикл получения документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      ТекДок = Док.ТекущийДокумент();
   КонецЦикла;
КонецЕсли;
```

См. также: ПолучитьДокумент, ОбратныйПорядок

ВыбратьПоНомеру

Открыть выборку документов по номеру.

Синтаксис:

```
ВыбратьПоНомеру(<Номер>, <Дата>, <ИдентВида>)
```

Англоязычный синоним:

```
SelectByNum
```

Параметры:

<Номер>	Строковое выражение, содержащее значение номера искомых документов.
<Дата>	Выражение типа «дата».
<ИдентВида>	Необязательный параметр. Строковое выражение, содержащее идентификатор вида документа или идентификатор Нумератора.

Возвращаемое значение:

Число 1 — если действие выполнено и в выборке есть хотя бы один документ; 0 — если действие не выполнено или в выборке нет ни одного документа.

Описание:

Метод ВыбратьПоНомеру открывает выборку всех документов с данным номером одного вида или одного нумератора (например, для поиска всех документов с совпадающими номерами).

В качестве второго параметра задается любая дата из диапазона, в котором нужно искать документ с данным номером. Поиск зависит от выбранного в конфигураторе способа уникальности номеров (по месяцу, году и др.).

Метод может быть использован для объекта Документ общего вида, тогда для поиска нужно указать в параметре <ИдентВида> идентификатор вида документа или идентификатор Нумератора.

Непосредственно сама выборка осуществляется при помощи метода ПолучитьДокумент в порядке возрастания даты и времени записи документов (если не задан обратный порядок методом ОбратныйПорядок).

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект("Документ");
   // открываем выборку документов
   Док.ВыбратьПоНомеру("Ав-0005", '01.01.98', "АктПереоценки");
   // цикл получения документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      ТекДок = Док.ТекущийДокумент();
   КонецЦикла;
КонецЕсли;
```

См. также: ПолучитьДокумент, ОбратныйПорядок

ВыбратьПоПоследовательюсти

Открыть выборку документов по заданной последовательности.

Синтаксис:

```
ВыбратьПоПоследовательности(<НачПериода>, <КонПериода> , <Последоват>)
```

Англоязычный синоним:

```
SelectBySequence
```

Параметры:

<НачПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки документов. Если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа.
<КонПериода >	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки документов. Если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом.
<Последоват>	Строковое выражение, содержащее идентификатор используемой Последовательности.

Возвращаемое значение:

Число 1 — если действие выполнено и в выборке есть хотя бы один документ; 0 — если действие не выполнено или в выборке нет ни одного документа.

Описание:

Метод ВыбратьПоПоследовательности открывает выборку документов по заданной последовательности.

Непосредственно сама выборка осуществляется при помощи метода ПолучитьДокумент в порядке возрастания даты и времени записи документов (если не задан обратный порядок методом ОбратныйПорядок).

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект ("Документ");
   // открываем выборку документов
   Док.ВыбратьПоПоследовательности(НачД, ПолучитьДокументТА(), "Упр");
   // цикл получения документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      ТекДок = Док.ТекущийДокумент();
   КонецЦикла;
КонецЕсли;
```

См. также: ПолучитьДокумент, ОбратныйПорядок

ОбратныйПорядок

Установить порядок выборки документов.

Синтаксис:

```
ОбратныйПорядок(<Режим>)
```

Англоязычный синоним:

```
BackwardOrder
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — выбирать документы в обратном порядке даты и времени; 0 — выбирать документы в порядке возрастания даты и времени. Значение по умолчанию — 1.

Возвращаемое значение:

Текущее значение порядка выборки документов (на момент до исполнения метода): 1 — обратный порядок выборки документов; 0 — выборка документов в порядке возрастания даты и времени.

Описание:

Метод ОбратныйПорядок устанавливает порядок выборки документов. Данный метод обычно используется до вызова одного из методов: ВыбратьДокументы, ВыбратьПодчиненныеДокументы, ВыбратьПоЗначению, который фактически открывает выборку. Дальнейшая выборка при помощи ПолучитьДокумент будет происходить в заданном порядке выборки.

По умолчанию, выборка документов выполняется в порядке возрастания даты и времени записи документов. Поэтому реально имеет смысл применять данный метод только в том случае, если надо получить обратный порядок выборки.

В тексте программного модуля возможно использование данного метода как процедуры или как функции. При использовании в качестве функции, возвращаемое значение соответствует текущему порядку выборки, которое было до вызова данного метода.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект("Документ");
   // устанавливаем обратный порядок выборки документов
   Док = ОбратныйПорядок(1);
   // открываем выборку документов
   Док.ВыбратьДокументы(ДатаНачало, ДатаКонец);
   // цикл получения документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
      ТекДок = Док.ТекущийДокумент();
   КонецЦикла;
КонецЕсли;
```

УстановитьФильтр

Назначить фильтр выборки документов.

Синтаксис:

```
УстановитьФильтр(<Проведенные>, <НеПроведенные>, <НеИмеющиеПризнаковУчета>,
                 <Оперативные>, <Расчетные>, <Бухг>)
```

Англоязычный синоним:

```
SetFilter
```

Параметры:

<Проведенные>	Числовое выражение: 0 — не включать в выборку проведенные документы; 1 — включать.
<НеПроведенные>	Числовое выражение: 0 — не включать в выборку непроведенные документы; 1 — включать.
<НеИмеющиеПризнаковУчета>	Числовое выражение: 0 — не включать в выборку документы не имеющие признаков учета; 1 — включать.
<Оперативные>	Числовое выражение: 0 — не включать в выборку оперативные документы; 1 — данный флаг не влияет на выборку; 2 — если оперативный документ, то включается в выборку.
<Расчетные>	Числовое выражение: 0 — не включать в выборку расчетные документы; 1 — данный флаг не влияет на выборку; 2 — если расчетный документ, то включается в выборку.
<Бухг>	Числовое выражение: 0 — не включать в выборку бухгалтерские документы; 1 — данный флаг не влияет на выборку; 2 — если бухгалтерский документ, то включается в выборку.

Описание:

Метод УстановитьФильтр назначает фильтр выборки документов.

Примеры:

- Док.УстановитьФильтр(1, 1, 1, 1, 1, 1) — в выборку включаются все документы;
- Док.УстановитьФильтр(1, 0, 1, 1, 1, 1) — в выборку включаются все проведенные документы;
- Док.УстановитьФильтр(1, 1, 0, 1, 0, 1) — в выборку включаются все документы, имеющие признак оперативный или бухгалтерский или оба, но не имеют признака расчетный;
- Док.УстановитьФильтр(1, 1, 1, 2, 1, 2) — в выборку включаются все документы, имеющие признаки и оперативный и бухгалтерский и те, которые не имеют признаков учета;
- Док.УстановитьФильтр(1, 1, 1, 2, 0, 2) — в выборку включаются все документы, имеющие признаки и оперативный и бухгалтерский, но не имеют признака расчета, и те, которые не имеют признаков учета;
- Док.УстановитьФильтр(1, 0, 0, 0, 1, 0) — в выборку не включаются проведенные документы, если они неоперативные и небухгалтерские и имеют признаки учета, т. е. фактически только проведенные расчетные.

Данный метод может использоваться только для документов, созданных функцией СоздатьОбъект.

Пример:

```
Док = СоздатьОбъект("Документ");
// устанавливаем фильтр выборки документов
Док.УстановитьФильтр(1, 0, 0, 1, 0, 0);
// открываем выборку документов
Док.ВыбратьДокументы(ДатаНачало, ДатаКонец);
// цикл получения документов
Пока Док.ПолучитьДокумент() > 0 Цикл
   ТекДок = Док.ТекущийДокумент();
КонецЦикла;
```

ПолучитьДокумент

Получить из выборки следующий документ.

Синтаксис:

```
ПолучитьДокумент()
```

Англоязычный синоним:

```
GetDocument
```

Возвращаемое значение:

Числовое значение: 1 — если следующий документ выбран успешно; 0 — если документ не найден (отсутствует).

Описание:

Метод ПолучитьДокумент выбирает следующий документ в последовательности выборки, открытой перед этим при помощи метода ВыбратьДокументы. Данный метод используется для организации цикла по документам.

Данный метод может использоваться только для позиционируемых объектов, созданных функцией СоздатьОбъект.

Пример:

```
Если Режим = "Подробный" Тогда
   Док = СоздатьОбъект("Документ");
   // Откроем выборку документов
   Док.ВыбратьДокументы(ДатаНачало, ДатаКонец);
   // Цикл выбора документов
   Пока Док.ПолучитьДокумент() > 0 Цикл
   // присвоим переменной ТекДок значение найденного документа
```

```
      ТекДок = Док.ТекущийДокумент();
      // ...
   КонецЦикла;
КонецЕсли;
```

См. также: ВыбратьДокументы

ВыбратьСтроки

Открыть выборку строк многострочной части документа.

Синтаксис:

```
ВыбратьСтроки()
```

Англоязычный синоним:

```
SelectLines
```

Возвращаемое значение:

Число: 1 — если действие выполнено и в выборке есть хотя бы одна строка; 0 — если действие не выполнено или в выборке нет ни одной строки.

Описание:

Метод ВыбратьСтроки открывает выборку строк многострочной части документа. Непосредственно сама выборка осуществляется при помощи метода ПолучитьСтроку в порядке возрастания номеров строк.

Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   // откроем выборку строк спецификации документа
   ВыбратьСтроки();
   // цикл получения строк спецификации документа
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

ПолучитьСтроку

Получить из выборки следующую строку табличной части документа.

Синтаксис:

```
ПолучитьСтроку()
```

Англоязычный синоним:

```
GetLine
```

Возвращаемое значение:

Число: 1 — если следующая строка табличной части документа выбрана успешно; 0 — если следующая строка документа не найдена (отсутствует).

Описание:

Метод ПолучитьСтроку выбирает следующую строку документа в последовательности выборки, открытой перед этим при помощи метода ВыбратьСтроки. Данный метод используется для организации цикла по строкам документа.

Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   // открываем выборку строк спецификации документа
   ВыбратьСтроки();
   // цикл выбора строк спецификации документа
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар=Товар;
      Регистр.ТовЗап.Количество=Количество;
      Регистр.ТовЗап.Стоимость=Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

Новый

Начать ввод нового документа.

Синтаксис:

```
Новый()
```

Англоязычный синоним:

```
New
```

Описание:

Метод `Новый` инициализирует создание нового документа. Собственно запись нового документа в информационную базу происходит при вызове метода `Записать`. После инициализации создания нового документа, как правило производится заполнение его реквизитов с последующим вызовом метода `Записать`.

Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.

Пример:

```
// данная процедура создает документ с № 1, в котором записывает
// все имеющиеся в справочнике должности
Процедура Оприходовать()
   Спр = СоздатьОбъект("Справочник.Должности");
   Док = СоздатьОбъект("Документ.Приказы");
   // найдем документ с № 1
   Док.НайтиПоНомеру("1", 0);
   Если Док.Выбран() = 0 Тогда
      // если документа с № 1 не существует, то создадим его
      Док.Новый();
      Док.НомерДок = "1";
      Док.ДатаДок='01.01.80';
      // запишем все должности в документ
      Спр.ВыбратьЭлементы();
      Пока Спр.ПолучитьЭлемент() > 0 Цикл
         Если (Док.Выбран() = 1) Тогда
            Док.НоваяСтрока();
            Док.Должность = Спр.ТекущийЭлемент();
         КонецЕсли;
      КонецЦикла;
      // запишем новый документ на диск
      Док.Записать();
   КонецЕсли;
КонецПроцедуры
```

ПрефиксНомера

Установить текущий префикс номера для документа.

Синтаксис:

```
ПрефиксНомера(<Префикс>)
```

Англоязычный синоним:

```
NumPrefix
```

Параметры:

`<Префикс>` Необязательный параметр. Строковое выражение — новый префикс номера для документа.

Возвращаемое значение:

Строковое значение текущего префикса документа (на момент до исполнения метода).

Описание:

Метод `ПрефиксНомера` устанавливает новый префикс для автоматического создания номера документа.

Данный метод может использоваться только для объектов, созданных Функцией `СоздатьОбъект` и в основном предназначен для использования при создании новых документов из языка с помощью метода `Новый`.

Пример:

```
ПрефиксНомера("Сч-");
```

См. также: `ПрефиксАвтоНумерации`, `УстановитьНовыйНомер`

УстановитьНовыйНомер

Установить новый номер с префиксом номера для документа.

Синтаксис:

```
УстановитьНовыйНомер(<Префикс>)
```

Англоязычный синоним:

SetNewNum

Параметры:

<Префикс>	Строковое выражение — префикс номера документа.

Описание:

Метод УстановитьНовыйНомер устанавливает новый номер документа с заданным префиксом <Префикс>. Данный метод производит корректное переприсвоение номера документа (с автоинкрементацией и резервированием номера на время ввода).

Пример:

```
УстановитьНовыйНомер("01-");
```

См. также: ПрефиксАвтоНумерации, ПрефиксНомера

НазначитьТип

Назначить тип для реквизита неопределенного вида.

Синтаксис:

```
НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
SetType
```

Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита документа неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту документа. Например: "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных. Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:

Метод НазначитьТип позволяет назначить тип для реквизита, которому в конфигураторе назначен тип «Неопределенный».

Пример:

```
Накладная.НазначитьТип("ТМЦ", "Справочник.Товары");
```

Записать

Записать документ.

Синтаксис:

```
Записать()
```

Англоязычный синоним:

```
Write
```

Описание:

Метод Записать выполняет запись в базу данных текущего нового или измененного документа.

Замечание. Если этот метод применяется в Модуле формы документа непосредственно к документу локального контекста, то данный метод отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Записать".

Пример:

```
Док.СоздатьОбъект("Документ.Заказ");
Док.АвтоВремяТекущее();
Док.Новый();
Док.УстановитьНовыйНомер("Прг-");
Док.Клиент = Константа.ОсновнойКлиент;
Док.Количество = 5;
Док.Записать();
Док.Провести(1, "Программно");
```

Удалить

Удалить документ.

Синтаксис:

удалить(<Режим>)

Англоязычный синоним:

Delete

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — непосредственное удаление; 0 — пометка на удаление. Значение по умолчанию — 1.

Описание:

Метод Удалить удаляет (или делает пометку на удаление) текущий документ. Данный метод разрешено применять только для тех документов, которые либо не проведены, либо они лежат за точкой ТА, т. е. предполагается, что предварительно интерактивно сдвинули ТА назад.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Замечание: Непосредственное удаление объекта следует применять очень аккуратно, так как это действие может нарушить ссылочную целостность информации. Данный режим не рекомендуется использовать, если на данный объект могут быть ссылки в других объектах, например в реквизитах существующих документов.

Пример:

```
Док = СоздатьОбъект("Документ.Приказы");
Если Док.Выбрать("Найди документ") > 0 Тогда
   Док.Удалить(1);
КонецЕсли;
```

См. также: ПометкаУдаления, СнятьПометкуУдаления

ПометкаУдаления

Прочитать флаг пометки удаления документа.

Синтаксис:

ПометкаУдаления()

Англоязычный синоним:

DeleteMark

Возвращаемое значение:

Число: 1 — если на документе стоит пометка удаления; 0 — если нет пометки удаления.

Описание:

Метод ПометкаУдаления позволяет прочитать значение пометки удаления документа.

Пример:

```
Процедура УсловноеУдаление(Докум)
   Док = СоздатьОбъект("Документ");
   Если Докум.Выбран() > 0 Тогда
      // позиционируем документ на переданном значении
      Док.НайтиДокумент(Докум);
      Если Док.Выбран() > 0 Тогда
         Если Вопрос("Удалять??? ", 1) = 1 Тогда
            Если Док.ПометкаУдаления() = 1 Тогда
               Предупреждение("Уже помечен на удаление!");
            Иначе
               // удаляем документ
               Док.Удалить(0);
            КонецЕсли;
         КонецЕсли;
      Иначе
         Предупреждение("Нечего удалять! ");
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: Удалить, СнятьПометкуУдаления

СнятьПометкуУдаления

Снять пометку удаления документа.

Синтаксис:

СнятьПометкуУдаления()

Англоязычный синоним:

ClearDeleteMark

Описание:
Метод СнятьПометкуУдаления снимает пометку удаления текущего документа.
Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.
Пример:
```
Процедура Восстановление(Докум)
   Док = СоздатьОбъект("Документ");
   Если Докум.Выбран() > 0 Тогда
      // позиционируем документ на переданном значении
      Док.НайтиДокумент(Докум);
      Если Док.Выбран() > 0 Тогда
         Если Док.ПометкаУдаления() = 1 Тогда
            // восстанавливаем
            Спр.СнятьПометкуУдаления();
         КонецЕсли;
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры _________ ________
```
См. также: Удалить, ПометкаУдаления

НоваяСтрока

Добавить новую строку в документ.
Синтаксис:
```
НоваяСтрока()
```
Англоязычный синоним:
```
NewLine
```
Описание:
Метод НоваяСтрока добавляет новую строку с очередным порядковым номером в многострочную часть документа.
Пример:
```
Процедура ОбработкаПодбора(Выб, КонФормы)
   Кол=0;
   Если ВвестиЧисло(Кол, "Введите количество", 10, 0) = 1 Тогда
      // добавим новую строку в спецификацию
      НоваяСтрока();
      Работа = Выб;
      Количество = Кол;
      АктивизироватьСтроку();
   КонецЕсли;
КонецПроцедуры
```

УдалитьСтроку

Удалить строку документа.
Синтаксис:
```
УдалитьСтроку()
```
Англоязычный синоним:
```
DeleteLine
```
Описание:
Метод УдалитьСтроку удаляет текущую строку из многострочной части Документа.
Пример:
```
Процедура УдалЗаданнойСтроки(Док, Стр)
   Если Док.Выбран() = 1 Тогда
      // если переданный параметр содержит значение документа, то
      // найдем в нем строку
      Док.ПолучитьСтрокуПоНомеру(Стр);
      // удалим найденную строку спецификации документа
      Док.УдалитьСтроку();
      Предупреждение("Документ не выбран!");
   КонецЕсли;
КонецЕсли;
```

УдалитьСтроки

Удалить все строки документа.

Синтаксис:
```
УдалитьСтроки()
```
Англоязычный синоним:
```
DeleteLines
```
Описание:
Метод `УдалитьСтроки` удаляет сразу все строки из многострочной части документа.
Пример:
```
УдалитьСтроки();
```

СортироватьСтроки

Сортировать многострочную часть документа по реквизитам.
Синтаксис:
```
СортироватьСтроки(<Колонки>)
```
Англоязычный синоним:
```
SortLines
```
Параметры:

`<Колонки>`	Перечисленные через запятую реквизиты многострочной части документа, по которым следует сортировать. Знаки "+", "-", предшествующие названию реквизита указывают направление сортировки по реквизиту. Знак "*" — сортировать по внутреннему значению реквизита.

Описание:
Метод `СортироватьСтроки` позволяет сортировать многострочную часть документа по реквизитам.
Пример:
```
Док = СоздатьОбъект("Документ.Приказы");
Если Док.Выбрать("Найди документ") > 0 Тогда
   Док.СортироватьСтроки("+Сумма, -НДС");
   Док.Записать ();
КонецЕсли;
```

ПолучитьПозицию

Получить позицию документа.
Синтаксис:
```
ПолучитьПозицию()
```
Англоязычный синоним:
```
GetPosition
```
Возвращаемое значение:
32-х символьное строковое значение позиции документа.
Описание:
Метод `ПолучитьПозицию` возвращает позицию документа.

Замечание. Позиция может существовать только у записанного документа.

Пример:
```
ПозицияДокумента = ВыбДокумент.ПолучитьПозицию();
```

ПринадлежитПоследовательности

Определить, принадлежит ли последовательности данный документ.
Синтаксис:
```
ПринадлежитПоследовательности(<Последов>, <НовоеЗначение>)
```
Англоязычный синоним:
```
BelongToSequence
```
Параметры:

`<Последов>`	Строковое выражение, определяющая Последовательность.
`<НовоеЗначение>`	Необязательный параметр. Число: 1 — документ принудительно устанавливается как принадлежащий последовательности; 0 — документ принудительно устанавливается как не принадлежащий последовательности. Использование этого параметра разрешено только при проведении документа (в Модуле документа).

Возвращаемое значение:
Число: 1 — если документ принадлежит последовательности; 0 — если не принадлежит.

Описание:

Метод ПринадлежитПоследовательности позволяет определить, принадлежит ли документ заданной последовательности.

Пример:

```
Если Док.ПринадлежитПоследовательности(«УпрУчет») = 1 Тогда
   Если Последовательность.УпрУчет.Сравнить(Док) = 1 Тогда
      Последовательность.УпрУчет.Установить(Док);
   КонецЕсли;
КонецЕсли;
```

ИспользоватьЖурнал

Использовать журнал для отбора документов и доступа к графам.

Синтаксис:

```
ИспользоватьЖурнал(<Журнал>, <ФлагОгрПросм>)
```

Англоязычный синоним:

```
UseJournal
```

Параметры:

<Журнал>	Идентификатор журнала, который предполагается использовать для отбора документов и доступа к графам.
<ФлагОгрПросм>	Необязательный параметр. Число: 1 — если выборка ограничивается документами, входящими в указанный журнал; 0 — если метод используется только для использования граф методом Графа. Значение по умолчанию 1.

Описание:

Метод ИспользоватьЖурнал позволяет при переборе документов командой Выбрать использовать журнал для отбора документов и доступа к графам журнала.

Пример:

```
Док = СоздатьОбъект("Документ.НаклПрих");
Док.ИспользоватьЖурнал("Приходные");
```

См. также: Графа

Графа

Выдает значение графы журнала для текущего документа.

Синтаксис:

```
Графа(<Графа>)
```

Англоязычный синоним:

```
Column
```

Параметры:

<Графа>	Идентификатор графы журнала, который задан методом ИспользоватьЖурнал.

Возвращаемое значение:

Значение заданной графы журнала для текущего документа.

Описание:

Метод Графа позволяет получить значение заданной графы журнала для текущего документа. Данный метод работает только в том случае, если ранее Применен метод ИспользоватьЖурнал.

Пример:

```
Док = СоздатьОбъект("Документ.НаклПрих");
Док.ИспользоватьЖурнал("Приходные", 0);
док.НайтиДокумент(ВыбДок);
Цена = Док.Графа("Цена");
```

См. также: ИспользоватьЖурнал

Блокировка

Установить/прочитать режим блокировки.

Синтаксис:

```
Блокировка(<ВклВыкл>)
```

Англоязычный синоним:

```
Locking
```

Параметры:

<ВклВыкл>	Необязательный параметр. Число: 1 — включить; 0 — выключить.

Возвращаемое значение:

Если при вызове метода параметр <ВклВыкл> не задан, то возвращается режим блокировки до выполнения метода. Число: 1 — заблокирован; 0 — свободен.

Если при вызове метода параметр <ВклВыкл> задан, то возвращается результат выполнения метода блокировки. Число: 1 — успешно; 0 — не получилось.

Описание:

Метод Блокировка позволяет установить/прочитать режим блокировки.

Пример:

```
Блокировка(0);
```

ПолучитьВремя

Прочитать время документа.

Синтаксис:

```
ПолучитьВремя(<Часы>, <Минуты>, <Секунды>)
```

Англоязычный синоним:

```
GetTime
```

Параметры:

<Часы>	Идентификатор переменной, в которую метод возвращает строковое значение минут записи документа.
<Секунды>	Идентификатор переменной, в которую метод возвращает строковое значение секунд записи документа.

Возвращаемое значение:

Строковое значение времени записи документа в виде "ЧЧ.ММ.СС".

Описание:

Метод ПолучитьВремя возвращает время документа в переданные для этого переменные <Часы>, <Минуты>, <Секунды>.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Функция ВремяДокумента(Док)
   Перем Ч;
   Перем М;
   Перем С;
   Если Док.Выбран() > 0 Тогда
      Док.ПолучитьВремя(Ч, М, С);
      Возврат "Документ записан в " + Ч + " час " + М + " мин. " + С + " с";
   Иначе
      Возврат "Документ не выбран!";
   КонецЕсли;
Конецфункции
```

УстановитьВремя

Установить время документа.

Синтаксис:

```
УстановитьВремя(<Часы>, <Минуты>, <Секунды>)
```

Англоязычный синоним:

```
SetTime
```

Параметры:

<Часы>	Числовое выражение.
<Минуты>	Числовое выражение.
<Секунды>	Числовое выражение.

Описание:

Метод УстановитьВремя устанавливает время документа в соответствии с заданными параметрами <Часы>, <Минуты>, <Секунды>.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:

```
Док = СоздатьОбъект("Документ.НаклПрих");
Если Док.Выбрать("Найди документ") > 0 Тогда
```

```
    Док.УстановитьВремя(13, 0, 0);
КонецЕсли;
```

См. также: ПриИзмененииВремениДокумента

АвтоВремяНачалоДня

Установить новому документу время на начало дня.

Синтаксис:

```
АвтоВремяНачалоДня()
```

Англоязычный синоним:

```
AutoTimeBegOfDay
```

Описание:

Метод АвтоВремяНачалоДня устанавливает режим, при котором новому документу записывается время на начало дня.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:

```
Процедура УстВремяДок(Док, Режим)
   Если Режим = "Начало" Тогда
      Док.АвтоВремяНачалоДня();
   ИначеЕсли Режим = "Конец" Тогда
      Док.АвтоВремяКонецДня();
   ИначеЕсли Режим = "Текущее" Тогда
      Док.АвтоВремяТекущее();
   Иначе
      Док.АвтоВремяОтключить();
   КонецЕсли;
КонецПроцедуры
```

АвтоВремяКонецДня

Установить новому документу время на конец дня.

Синтаксис:

```
АвтоВремяКонецДня()
```

Англоязычный синоним:

```
AutoTimeEndOfDay
```

Описание:

Метод АвтоВремяКонецДня устанавливает режим, при котором новому документу записывается время на конец дня.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:

См. предыдущий пример

АвтоВремяТекущее

Установить новому документу текущее время.

Синтаксис:

```
АвтоВремяТекущее()
```

Англоязычный синоним:

```
AutoTimeCurrent
```

Описание:

Метод АвтоВремяТекущее устанавливает режим, при котором новому документу записывается текущее время.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:
```
Процедура УстВремяДок(Док, Режим)
   Если Режим = "Начало" Тогда
      Док.АвтоВремяНачалоДня();
   ИначеЕсли Режим = "Конец" Тогда
      Док.АвтоВремяКонецДня();
   ИначеЕсли Режим = "Текущее" Тогда
      Док.АвтоВремяТекущее();
   Иначе
      Док.АвтоВремяОтключить();
   КонецЕсли;
КонецПроцедуры
```

АвтоВремяПослеТА

Установить новому документу время после ТА.

Синтаксис:
```
АвтоВремяПослеТА()
```
Англоязычный синоним:
```
AutoTimeAfterAP
```
Описание:

Метод `АвтоВремяПослеТА` устанавливает режим, при котором, если совпадает дата документа с датой ТА, то новому документу автоматически записывается время после ТА. Метод исполняется только если установлена компонента оперативный учет.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:
```
Процедура УстВремяДок(Док, Режим)
   Если Режим = "Начало" Тогда
      Док.АвтоВремяНачалоДня();
   ИначеЕсли Режим = "Конец" Тогда
      Док.АвтоВремяКонецДня();
   ИначеЕсли Режим = "Текущее" Тогда
      Док.АвтоВремяТекущее();
   ИначеЕсли Режим = "ПослеТА" Тогда
      Док.АвтоВремяПослеТА();
   Иначе
      Док.АвтоВремяОтключить();
   КокецЕсли;
КонецПроцедуры
```

АвтоВремяОтключить

Отключает режим автоматической установки времени документа.

Синтаксис:
```
АвтоВремяОтключить()
```
Англоязычный синоним:
```
AutoTimeOff
```
Описание:

Метод `АвтоВремяОтключить` отключает режим автоматической установки времени нового документа, при этом новые документы заносятся с временем устанавливаемым при помощи метода `УстановитьВремя`.

Замечание. Для новых документов в модуле формы доступно использование методов установки признака автоформирования времени документа и собственно установки времени. Если эти методы использованы, то диалог запроса времени не вызывается и игнорируются пользовательские установки в параметрах системы.

Пример:

См. предыдущий пример

См. также: `УстановитьВремя`

Провести

Выполнить проведение документа.

Синтаксис:

```
Провести(<Режим>, <Знач>)
```

Англоязычный синоним:

```
MakeActions
```

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 0 — проводить документ без сдвига ТА; 1 — проводить непроведенный документ реальным временем (со сдвигом ТА); 2 — перепроводить проведенный документ реальным временем (со сдвигом ТА); 3 —проводить любой (непроведенный, проведенный) документ реальным временем (со сдвигом ТА). Значение по умолчанию — 0.

Данный параметр устанавливает режим проведения документа после ТА. Этот метод относится к случаю проведения документа в пределах даты, в которой находится ТА. Дело в том, что при записи документа, документ кроме даты получает еще и время документа. В многопользовательском режиме часто возникает ситуация, когда в момент проведения документа, он уже оказывается не последним в потоке проводимых документов (провели документ с другого рабочего места), в этом случае система 1С:Предприятие дает на выбор два варианта проведения:

- провести документ тем временем, каким он был записан, т. е. фактически задним временем (до ТА);
- изменить время документа в рамках текущей даты, чтобы документ стал самым последним и проводился в потоке, т. е. в ТА.

<Знач> Необязательный параметр. Выражение произвольного типа. Значение данного параметра будет передано системой в качестве параметра при запуске предопределенной процедуры ОбработкаПроведения. Использовать данное значение можно, например, для того, чтобы в процедуре ОбработкаПроведения правильно отработать режим проведения, т. к. это программный, а не интерактивный и не групповой (см. ГрупповаяОбработка) способ проведения. По умолчанию передается пустое значение.

Возвращаемое значение:

Число: 1 — проведение документа завершено успешно; 0 — проведение документа не выполнено.

Описание:

Метод Провести запускает процедуру проведения документа. Возвращаемое значение можно использовать, например, в качестве условия принятия решения при завершении обработки транзакции.

Замечание. Данный метод нельзя использовать в теле предопределенной процедуры ОбработкаПроведения.

Замечание. Если этот метод применяется в Модуле формы документа непосредственно к документу локального контекста, то данный метод отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Провести". В этом случае, если параметр <Режим> опущен, то документ проводится в режиме, соответствующем установкам системы меню «Сервис» - «Параметры».

Пример:

```
Док.СоздатьОбъект("Документ.Заказ");
Док.АвтоВремяТекущее();
Док.Новый();
Док.УстановитьНовыйНомер("Прг-");
Док.Клиент = Константа.ОсновнойКлиент;
Док.Количество = 5;
Док.Записать();
Док.Провести(1, "Программно");
```

См. также: ОбработкаПроведения, ГруппровыяОбработка

СделатьНеПроведенным

Отменить проведение документа.

Синтаксис:

```
СделатьНеПроведенным()
```

Англоязычный синоним:

```
UnPost
```

Описание:

Метод СделатьНеПроведенным отменяет проведение документа. Данный метод нельзя использовать в теле предопределенной процедуры ОбработкаПроведения.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:

```
Док.СоздатьОбъект("Документ.Заказ");
Док.НайтиДокумент(Докум);
Док.СделатьНеПроведенным();
```

См. также: Провести, ПриОтменеПроведенияДокумента

СравнитьТА

Сравнить дату и время документа с Точкой актуальности итогов.

Синтаксис:

```
СравнитьТА()
```

Англоязычный синоним:

```
CompareWithAP
```

Возвращаемое значение:

Числовое значение:

- 1(плюс единица) — если дата и время документа больше даты и времени Точки актуальности итогов.
- 0 — если дата и время документа равно дате и времени Точки актуальности итогов.
- -1 (минус единица) — если дата и время документа меньше даты и времени Точки актуальности итогов.
- -2 (минус два) в случае, если новый документ, который сравнивается с ТА, еще не записан, то есть он еще не имеет никакой позиции на оси времени.

Описание:

Метод СравнитьТА сверяет дату и время документа с датой и временем Точки актуальности итогов, позволяя определить положение на оси времени момента записи документа по отношению к текущему положению ТА.

Пример:

```
Процедура ОбработкаПроведения()
   Если СравнитьТА() > 0 Тогда
      // если документ после ТА, вызываем процедуру
      ПроводкаДокументаБудущимЧислом();
   ИначеЕсли СравнитьТА() < 0 Тогда
      // если документ до ТА, вызываем процедуру
      ПроводкаДокументаЗаднимЧислом();
   Иначе
      // если документ в ТА, вызываем процедуру
      ПроводкаДокумента();
   КонецЕсли;
КонецПроцедуры
```

СуществуетОперация

Устанавливает (возвращает) признак существования операции у документа.

Синтаксис:

```
СуществуетОперация(<Признак>)
```

Англоязычный синоним:

```
OperExist
```

Параметры:

<Признак>	Числовое выражение: 1 — существует операция, 0 — не существует операции.

Возвращаемое значение:

Числовое значение — текущее значение признака: 1 — у документа существует операция; 0 — у документа не существует операции.

Описание:

В метаданных для конкретного вида документа (имеющего признак «Бухгалтерский учет») устанавливается режим записи операции. Операция может записываться для документов данного вида в режимах: «Всегда», «Выборочно» и «Только при проведении».

Метод СуществуетОперация применяется в случае использования режима «Выборочно». Он устанавливает признак наличия операции у конкретного документа. Данная возможность позволяет записывать операцию для документов конкретного вида в зависимости от некоторых условий. Для режимов записи операции «Всегда» и «Только при проведении» данный метод смысла не имеет.

Пример:

* Для накладной операция будет записываться только если отгрузка ведется с основного склада.

```
Процедура ПриЗаписи()
```

```
   Если Склад = ОснСклад Тогда
      СуществуетОперация(1);
   Иначе
      СуществуетОперация(0);
   КонецЕсли;
КонецПроцедуры
```

ВыгрузитьТабличнуюЧасть

Выгрузить многострочную часть документа в таблицу значений.

Синтаксис:

```
ВыгрузитьТабличнуюЧасть(<Знач>, <Реквизиты>)
```

Англоязычный синоним:

```
UnloadTable
```

Параметры:

<Знач>	Идентификатор переменной, содержащей значение типа «Таблица значений» или «Список значений», куда нужно выгрузить данные из многострочной части документа. Если переданное значение пустое, то система сама создаст объект типа «Таблица значений». Колонки совмещаются по идентификаторам.
<Реквизиты>	Необязательный параметр. Список реквизитов многострочной части документа через запятую. Данный параметр указывает, какие реквизиты выгружать, если не указано то все. Для номера строки документа в таблице значений создается отдельная колонка с идентификатором "НомерСтрокиДокумента". При задании какие колонки выгружать, для выгрузки номера строки надо указать колонку "НомерСтроки".

Описание:

Метод ВыгрузитьТабличнуюЧасть выгружает многострочную часть документа в таблицу значений или список значений. Если в качестве значения для выгрузки задан список значений, то система выгружает данные из многострочной части документа последовательно по реквизитам.

Пример:

```
ВыбДокум.ВыгрузитьТабличнуюЧасть(ТаблЗнач, "Товар, Сумма, НДС, Цена");
```

ЗагрузитьТабличнуюЧасть

Загрузить многострочную часть документа из таблицы значений.

Синтаксис:

```
ЗагрузитьТабличнуюЧасть(<ТаблЗнач>)
```

Англоязычный синоним:

```
LoadTable
```

Параметры:

<ТаблЗнач>	Таблица значений, откуда загружается многострочная часть документа. Колонки совмещаются по идентификаторам.

Описание:

Метод ЗагрузитьТабличнуюЧасть загружает многострочную часть документа из таблицы значений.

Пример:

```
ВыбДокум.ЗагрузитьТабличнуюЧасть(ТаблЗнач);
```

Методы контекста Модуля формы документа

Описанные в данном разделе методы доступны только в контексте Модуля формы документа (см. «Виды программных модулей»).

ПриЗаписиПерепроводить

Установить режим обязательного перепроведения документа при записи.

Синтаксис:

```
ПриЗаписиПерепроводить(<Режтл>)
```

Англоязычный синоним:

```
ReMakeActionsWhenWrite
```

Параметры:

<Режим>	Числовое выражение: 1 — устанавливает флаг обязательного перепроведения при записи уже проведенного документа; 0 — снимает флаг обязательного перепроведения при записи уже проведенного документа.

Описание:

Метод ПриЗаписиПерепроводить устанавливает режим обязательного перепроведения при записи ранее проведенного документа (интерактивный отказ от проведения документа или ошибка модуля приведет к отмене записи отредактированного документа).

Данный метод доступен только в контексте Модуле формы документа (см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы Пример:

```
Процедура ПриОткрытии()
   ПриЗаписиПерепроводить(1);
КонецПроцедуры
```

ПроводитьПослеТА

Установить режим проведения документа после ТА.

Синтаксис:

ПроводитьПослеТА(<ФлагДляНеПровДокумента>, <ФлагДляПровДокумента>)

Англоязычный синоним:

PostingAfterTA

Параметры:

<ФлагДляНеПровДокумента>	Режим проведения документа после ТА. Число: -1 (минус единица) — проводить документ всегда задним числом; 0 — при проведении запрашивать режим проведения документа; 1 — проводить документ в реальном потоке времени, т. е. при проведении время документа автоматически устанавливается на время после ТА.
<ФлагДляПровДокумента>	Режим перепроведения документа после ТА. Числовое выражение: -1 (минус единица) — проводить документ всегда задним числом; 1 — проводить документ в потоке.

Возвращаемое значение:

Текущее значение режим перепроведения документа в зависимости от про-веденности.

Описание:

Метод ПроводитьПослеТА устанавливает режим проведения документа после ТА. Этот метод относится к случаю проведения документа в пределах даты, в которой находится ТА. Дело в том, что при записи документа, документ кроме даты получает еще и время документа. В многопользовательском режиме часто возникает ситуация, когда в момент проведения документа, он уже оказывается не последним в потоке проводимых документов (провели документ с другого рабочего места), в этом случае система ЮПредприятие дает на выбор три варианта проведения:

- провести документ тем временем, каким он был записан, т. е. фактически задним временем (до ТА);
- изменить время документа (в рамках текущей даты), чтобы документ стал самым последним и проводился в потоке, т. е. после ТА;
- запросить у пользователя, в каком режиме провести документ.

Данный метод доступен только в контексте Модуле формы документа (см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы документа.

Пример:

```
Процедура ПриОткрытии()
   ПроводитьПослеТА(1, 1);
КонецПроцедуры
```

АктивизироватьСтроку

Установить курсор на указанной строке многострочной части документа.

Синтаксис:

АктивизироватьСтроку(<НомСтроки>)

Англоязычный синоним:

ActivateLine

Параметры:

<НомСтроки>	Необязательный параметр. Номер строки, которую следует активизировать. Если параметр опущен, то активизируется текущая строка многострочной части документа.

Описание:

Данный метод в основном предназначен для использования в системной предопределенной процедуре ОбработкаПодбора. Метод АктивизироватьСтроку выполняет визуальную активизацию строки документа в форме редактирования документа после выхода из предопределенной процедуры ОбработкаПодбора.

Данный метод доступен только в контексте Модуле формы документа (см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы документа.

Пример:
```
Процедура ОбработкаПодбора(Выб, КонтФормы)
   Кол = 0;
   Если ВвестиЧисло(Кол, "Введите количество", 10, 0) = 1 Тогда
      НоваяСтрока();
      Товар = Выб;
      УстанЦеныРасх(Контекст);
      Количество = Кол;
      Выч_суммы_накл(Контекст);
      АктивизироватьСтроку();
      Активизировать("Цена");
   КонецЕсли;
КонецПроцедуры
```
См. также: ОбработкаПодбора

ИзменениеПорядкаСтрок

Установить режим изменения порядка строк в форме документа.

Синтаксис:
```
ИзменениеПорядкаСтрок(<Разрешить>)
```
Англоязычный синоним:
```
ChangeLinesOrder
```
Параметры:

<Разрешить>	Необязательный параметр. Число: 1 — разрешить изменение порядка строк в документе; 0 — запретить. Если параметр не задан, то режим не меняется.

Возвращаемое значение:

Режим изменения порядка строк до исполнения метода. Число: 1 — разрешено изменение порядка строк в документе; 0 — запрещено.

Описание:

Метод ИзменениеПорядкаСтрок позволяет установить режим изменения порядка строк в форме документа.

Данный метод доступен только в контексте Модуле формы документа (см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы документа.

Пример:
```
ИзменениеПорядкаСтрок(0);
```

Модифицированность

Возвратить признак изменения реквизитов текущей формы документа.

Синтаксис:
```
Модифицированность()
```
Англоязычный синоним:
```
Modify
```
Возвращаемое значение:

Числовое значение: 1 — если реквизиты текущей формы документа были изменены; 0 — если нет.

Описание:

Метод Модифицированность возвращает признак изменения реквизитов формы (в режиме исполнения он отображается символом (*) в заголовке окна формы).

Данный метод доступен только в Модуле формы документа(см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы документа.

Пример:
```
Изм = Модифицированность();
```

Предопределенные процедуры Модуля формы документа

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях формы документов (см. «Виды программных модулей»).

В основном данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении

соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ВводНового

Предопределенная процедура при вводе нового документа.

Синтаксис:

```
ВводНового(<ПризнКопирования>, <ОбъектКопирования>)
```

Англоязычный синоним:

```
InputNew
```

Параметры:

<ПризнКопирования>	Признак того, что объект введен копированием. Число: 1 — объект введен копированием, 0 — просто новый объект. Данный признак может быть использован для анализа необходимости инициализации реквизитов нового объекта.
<ОбъектКопирования>	Объект, который был скопирован.

Описание:

Вызов процедуры ВводНового производится в системе 1С:Предприятие неявно в момент выбора пункта меню «Действия» — «Новый» при работе с Документами. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов нового документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя вводить документы), ввода нового документа и открытие его формы не будет выполнено.

Процедуру ВводНового контекста документа можно размещать в Модулях форм документов (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНового();
   Валюта = Константа.ДефВалютаПродажи;
   Валюта.ИспользоватьДату(ДатаДок);
   Дата_Курса = ДатаДок;
   Если Число(Валюта.Код) = Число(Константа.БазоваяВалюта.Код) Тогда
      //У базовой валюты курс = 1, поэтому отображаем курс Основной валюты
      Константа.ОсновнаяВалюта.ИспользоватьДату(ДатаДок);
      Курс = Константа.ОсновнаяВалюта.Текущ_курс;
   Иначе
      Курс = Валюта.Текущ_курс;
   КонецЕсли;
   Фирма = Константа.ДефФирма;
   Клиент = Константа.ДефПокупатель;
КонецПроцедуры
```

См. также: СтатусВозврата

ВводНаОсновании

Предопределенная процедура при вводе нового документа на основании.

Синтаксис:

```
ВводНаОсновании(<ДокументОснование>)
```

Англоязычный синоним:

```
InputCausedBy
```

Параметры:

<ДокументОснование>	Значение документа, на основании которого вводится новый документ.

Описание:

Вызов процедуры ВводНаОсновании производится в системе 1С:Предприятие неявно после выбора пункта меню «Действия» - «Ввести на основании». В этот момент система подставляет фактическое значение параметра <ДокументОснование>, содержащее документ, на котором находился курсор в момент выполнения данной команды.

Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов нового документа, вводимого на основании другого. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя вводить новые документы), ввода нового документа и открытие его формы не будет выполнено.

Процедуру ВводНаОсновании можно размещать только в Модулях форм документов (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНаОсновании(ДокОснование)
   Если (ДокОснование.Вид() = "ПриходнаяРеализ")
         ИЛИ ДокОснование.Вид() = "РасходнаяРеализ") Тогда
```

```
      ПризнакРеализатора = Перечисление.ПризнакиРеализаторов.Реализатор_Клиент;
      ДатаДок = РабочаяДата();
      Валюта = Константа.БазоваяВалюта;
      Валюта.ИспользоватьДату(ДатаДок);
      Дата_Курса = ДокОснование.Дата_Курса;
      Курс = ДокОснование.Курс;
      НДС = ДокОснование.НДС;
      СН = ДокОснование.СН;
      Фирма = ДокОснование.Фирма;
      Основание = "" + ДокОснование + " от " + ДокОснование.ДатаДок;
      ДокОснование.ВыбратьСтроки();
      Пока ДокОснование.ПолучитьСтроку() = 1 Цикл
         НоваяСтрока();
         Товар = ДокОснование.Товар;
         Цена = ДокОснование.Цена;
         Единица = ДокОснование.Единица;
         Коэффициент = ДокОснование.Коэффициент;
         Количество = ДокОснование.Количество;
         Сумма = ДокОснование.Сумма;
      КонецЦикла;
   Иначе
      Предупреждение("Этот Документ можно вводить только на основании
                     |накладных по реализации!");
      ВводНового();
   КонецЕсли;
   Валюта_Прежн = Валюта;
   Курс_Прежн = Курс;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура при записи документа.

Синтаксис:

ПриЗаписи()

Англоязычный синоним:

OnWrite

Описание:

Вызов предопределенной процедуры ПриЗаписи производится в системе 1С:Предприятие при интерактивной записи документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данно-wy пользователю нельзя изменять реквизиты документа), запись документа не будет выполнена.

Данная предопределенная процедура может располагаться в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриЗаписи()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права изменять документ!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриНачалеРедактированияСтроки

Предопределенная процедура при редактировании существующей строки документа.

Синтаксис:

ПриНачалеРедактированияСтроки()

Англоязычный синоним:

OnStartEditLine

Описание:

Вызов предопределенной процедуры ПриНачалеРедактированияСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования существующей строки многострочной части документа. Если в данной

предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять документ), то запись не будет изменена.

Данная предопределенная процедура может располагаться в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права изменит, документ 1", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВводеСтроки

Предопределенная процедура при вводе новой строки многострочной части документа.

Синтаксис:

```
ПриВводеСтроки()
```

Англоязычный синоним:

```
OnNewLine
```

Описание:

Вызов предопределенной процедуры ПриВводеСтроки производится в системе 1С:Предприятие при интерактивном вводе новой строки многострочной части документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять документ), то новая строка не будет инициирована.

Данная предопределенная процедура может располагаться в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриВводеСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права добавлять строки!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриРедактированииНовойСтроки

Предопределенная процедура при редактировании новой строки многострочной части документа.

Синтаксис:

```
ПриРедактированииНовойСтроки()
```

Англоязычный синоним:

```
OnEditNewLine
```

Описание:

Вызов предопределенной процедуры ПриРедактированииНовойСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования новой строки многострочной части документа. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов строки табличной части документа. В данной предопределенной процедуре установка статуса возврата не имеет смысла, т. к. отказаться от ввода новой строки в этот момент уже невозможно.

Данная предопределенная процедура может располагаться в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриРедактированииНовойСтроки()
   Количество = 10;
   Коэффициент = 1;
   Скидка=0;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриОкончанииРедактированияСтроки

Предопределенная процедура при окончании редактирования строки многострочной части документа.

Синтаксис:

```
ПриОкончанииРедактированияСтроки(<НовСтр>)
```

Англоязычный синоним:

OnFinishLineEdit

Параметры:

<НовСтр> Флаг новой строки. Число: 1 — если произошло окончание ввода новой строки, 0 — если произошло окончание редактирования существующей строки.

Описание:

Вызов предопределенной процедуры ПриОкончанииРедактированияСтроки производится в системе 1С:Предприятие в момент окончания интерактивного редактирования строки многострочной части документа. Данная процедура может использоваться, например, для проверки значений (по умолчанию) реквизитов строки табличной части документа. В данной предопределенной процедуре установка статуса возврата не имеет смысла, т. к. отказаться от ввода строки в этот момент уже невозможно.

Замечание. Данный метод не вызывается при отказе от ввода новой строки по клавише <Esc>. Однако, метод вызывается при нажатии клавиши <Esc> во время редактирования существующей строки, но при этом последнее значение текущего редактируемого с клавиатуры поля не доступно, т. к. от его значения в данном случае отказываются, т. е. строка доступна в том виде, в котором она собственно и остается в документе.

Данная предопределенная процедура может располагаться в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриОкончанииРедактированияСтроки()
   Если Количество = 0 Тогда
      Количество = 1;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУдаленииСтроки

Предопределенная процедура при удалении строки многострочной части документа.

Синтаксис:

ПриУдаленииСтроки()

Англоязычный синоним:

OnDeleteLine

Описание:

Вызов предопределенной процедуры ПриУдаленииСтроки производится в системе 1С:Предприятие при интерактивном удалении строки многострочной части документа. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять документ), удаление строки документа не будет выполнено.

Данная предопределенная процедура может располагаться только в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриУдаленииСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права удалять строки!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриИзмененииПорядкаСтрок

Предопределенная процедура при изменении порядка строк многострочной части документа.

Синтаксис:

ПриИзмененииПорядкаСтрок(<Действие>)

Англоязычный синоним:

OnChangeLinesOrder

Параметры:

<Действие> Число: -1 — перемещение строки вверх; 1 — перемещение строки вниз; 0 — перенумерация строк.

Описание:

Вызов предопределенной процедуры ПриИзмененииПорядкаСтрок произ-^дится в системе 1С:Предприятие при интерактивном изменении порядка строк многострочной части документа (перемещения строк вверх-вниз, перену-

мерация, сортировка). Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять документ), изменении порядка строк документа не будет выполнено.

Данная предопределенная процедура может располагаться только в Модулях формы документов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриИзмененииПорядкаСтрок()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права изменять документ!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Методы контекста Модуля документа

Описанные в данном разделе методы доступны только в контексте Модуля документа (см. «Виды программных модулей»).

ИтогиАктуальны

Возвратить флаг актуальности итогов.

Синтаксис:

ИтогиАктуальны()

Англоязычный синоним:

AreTotalsActual

Возвращаемое значение:

Числовое значение: 1 — итоги актуальны, 0 — нет.

Описание:

Метод ИтогиАктуальны позволяет определить — выполняется ли проведение документа в реальном времени или задним числом (когда нельзя обращаться к текущим остаткам регистров).

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Системные предопределенные процедуры»).

Пример:

```
Процедура ОбработкаПроведения()
   // Для расходной накладной ВыбратьСтроки();
   Пока ПолучитьСтроку() > 0 Цикл
      Если ИтогиАктуальны() > 0 Тогда
         // если итоги актуальны,
         // то текущие стоимости берем из текущих остатков
         Регистр.ОстаткиТоваров.Склад = Скпад;
         Регистр.ОстаткиТоваров.Товар = Товар;
         Регистр.ОстаткиТоваров.ОстаткиПолучить();
         Регистр.ОстаткиТоваров.БазоваяСтоимость =
                 Регистр.ОстаткиТоваров.БазоваяСтоимость * Количество *
                 Коэффициент / Регистр.ОстаткиТоваров.ОстатокТовара;
         Регистр.ОстаткиТоваров.ВалютнаяСтоимость =
                 Регистр.ОстаткиТоваров.ВалютнаяСтоимость * Количество *
                 Коэффициент / Регистр.ОстаткиТоваров.ОстатокТовара;
         Регистр.ОстаткиТоваров.ОстатокТовара = Количество * Коэффициент;
         Регистр.ОстаткиТоваров.ДвижениеРасходВыполнить();
      Иначе
         // если итоги не актуальны,
         //то стоимости берем из врем. расчета Регистра
         Рег = СоздатьОбъект("Регистр.ОстаткиТоваров");
         Рег.ВременныйРасчет();
         РассчитатьРегистрыНа(ТекущийДокумент());
         Рег.Товар = Товар;
         Рег.Склад = Склад;
         Рег.ОстаткиПолучить();
         Регистр.ОстаткиТоваров.БазоваяСтоимость =
                 Рег.БазоваяСтоимость * Количество * Коэффициент /
                 Рег.ОстатокТовара;
         Регистр.ОстаткиТоваров.ВалютнаяСтоимость =
```

```
                Рег.ВалютнаяСтоимость * Количество * Коэффициент /
                Рег.ОстатокТовара;
        Регистр.ОстаткиТоваров.ОстатокТовара = Количество * Коэффициент;
        Регистр.ОстаткиТоваров.Склад = Склад;
        Регистр.ОстаткиТоваров.Товар = Товар;
        Регистр.ОстаткиТоваров.ДвижениеРасходВыполнить();
    КонецЕсли;
  КонецЦикла;
КонецПроцедуры
```

ГрупповаяОбработка

Возвратить флаг групповой обработки.

Синтаксис:

```
ГрупповаяОбработка()
```

Англоязычный синоним:

```
IsGroupProcessing
```

Возвращаемое значение:

Число: 1 — групповое обработка, 0 — проведение документа по одному.

Описание:

Метод `ГрупповаяОбработка` позволяет определить, выполняется ли проведение документа группой (при помощи режима «Управление итогами») или по одному (интерактивно).

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре `ОбработкаПроведения` (см. гл. «Системные предопределенные процедуры»).

Пример:

```
Лроцедура ОбработкаПроведения()
  Если ГрупповаяОбработка() > 0 Тогда
    // если групповая проводка документов, то вызываем процедуру
    ПроводкаДокументаГруппой();
  Иначе
    // если не групповая проводка документов, то
    ПроводкаДокументаОдиночного();
  КонецЕсли;
КонецПроцедуры
```

См. также: `ОбработкаПроведения`, `Провести`

НеПроводитьДокумент

Отменить процесс проведения документа.

Синтаксис:

```
НеПроводитьДокумент()
```

Англоязычный синоним:

```
DoNotTransactDoc
```

Описание:

Метод `НеПроводитьДокумент` фактически отменяет весь процесс проведения документа (отменяет транзакцию проведения). Вызов данного метода приводит к тому, что все движения регистров, вызванные в предопределенной процедуре `ОбработкаПроведения`, будут проигнорированы и документ не будет проведен. Метод `НеПроводитьДокумент` не завершает процедуру `ОбработкаПроведения` (для завершения процедуры нужно отдельно вызвать `Возврат`).

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре `ОбработкаПроведения` (см. Гл. «Системные предопределенные процедуры»).

По своему действию данный метод полностью аналогичен вызову системной функции `СтатусВозврата` с параметром 0, т. е. отменить действие.

Пример:

```
Процедура ОбработкаПроведения();
  Если Число(Клиент.Выбран()) > 0 Тогда
    ДвиженияРегистраРеализации();
  Иначе
    Сообщить("Не выбран клиент по реализации!!! Документ не проводится!!!");
    НеПроводитьДокумент();
    Возврат;
  КонецЕсли;
КонецПроцедуры
```

См. также:

```
СтатусВозврата
```

УстановитьРеквизитСправочника

Записать значение периодического реквизита справочника с привязкой к документу.

Синтаксис:

```
УстановитьРеквизитСправочника(<ЭлементСправочника>, <НазваниеРеквизита>,
                  <Значение>, <ДатаУстановки>, <ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
SetRefField
```

Параметры:

<ЭлементСправочника>	Выражение, задающее элемент справочника, в котором требуется записать новое значение периодического реквизита.
<НазваниеРеквизита>	Строковое выражение — название периодического реквизита справочника (как он назван в конфигураторе) в котором требуется записать новое значение периодического реквизита.
<Значение>	Новое значение периодического реквизита.
<ДатаУстановки>	Выражение типа «дата», на которую требуется установить новое значение периодического реквизита. Этот параметр имеет смысл только для не оперативных документов и позволяет установить значение на другую дату чем дата документа, но тогда с пустым временем.
<ИмяТипа>	Необязательный параметр. Строковое выражение — название типа данных (или Вид субконто).
<Длина>	Необязательный параметр. Число — длина числового или строкового значения.
<Точность>	Необязательный параметр. Число знаков после десятичной точки для числового значения.

Описание:

Метод УстановитьРеквизитСправочника записывает новое значение периодического реквизита справочника с привязкой к документу. Использование Данного метода позволяет вносить изменения в справочники с привязкой к проведению документа. Это значит, что при удалении или редактировании или отказе от проведения документа все сделанные им изменения в справочнике будут корректно отменены. Средства языка позволяют выбирать такие движе-ния справочника по документу или определять документ, который произвел изменения в справочнике (см. «Работа с объектом Периодический»).

Параметры <ИмяТипа>, <Длина> и <Точность> следует указывать при установке значения периодического реквизита справочника имеющего неопределенный тип.

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Системные предопределенные процедуры»).

Пример:

```
// при работе в контексте документа, его реквизит "Сотрудник"
// является переменной типа "справочник",
// чтобы установить новое значение периодического
// реквизита "Оклад" этого элемента
// справочника с привязкой к документу
Процедура ОбработкаПроведения();
   // .....
   Моклад = Константа.МаксимальныйОклад;
   УстановитьРеквизитСправочника(Сотрудник, "Оклад", МОклад, ДатаДок);
   // .....
КонецПроцедуры
```

ОчиститьДвижения

Удалить движения документа.

Синтаксис:

```
ОчиститьДвижения(<ВидыДвижений>)
```

Англоязычный синоним:

```
ClearActions
```

Параметры:

<ВидыДвижений>	Необязательный параметр. Строковое выражение, в котором через «;» перечислены виды движений. В качестве видов движений можно задавать следующие строковые значения: • "Регистр.XXXXXX" — движение конкретного регистра, где XXXXXX — конкретный вид регистра; • "Операция" — бухгалтерская операция;

- "Справочник" — все изменения периодических реквизитов справочников;
- "ЖурналРасчетов" — все измененияв журналах расчетов. Если данный параметр не задан, то производится полная очистка всех движений.

Описание:

Метод ОчиститьДвижения удаляет существующие движения документа.

Если метод не вызывается при проведении документа, то существующие движения документа не стираются вообще.

Если метод не вызывается при распроведении документа, то существующие движения после распроведения стираются.

Использование данного метода позволяет:

- допроводить документ, т. е. доделывать движения в добавок к уже существующим;
- делать движения, анализируя ранее сделанные и уже существующие;
- выполнять распроведение документа, анализируя существующие движения;
- не очищать существующие движения при перепроведении, если это не нужно.

Замечание. Данный метод доступен только при отключенном в конфигурации флаге «Автоматическое удаление движений».

Замечание. Данный метод доступен только в контексте Модуле формы документа (см. «Виды программных модулей»). Действие данного метода относится только к текущему документу, который доступен в локальном контексте Модуля формы документа.

Пример:

```
Процедура ОбработкаПроведения()
   ОчиститьДвижения("Операция; Справочник; ЖурналРасчетов");
   ДвиженияРегистровНакладных(Контекст);
КонецПроцедуры
// ————————————————————————————————
Процедура ОбработкаУдаленияПроведения()
   ОчиститьДвижения();
КонецПроцедуры
```

ПривязыватьСтроку

Записывать номер строки документа в движениях документа.

Синтаксис:

ПривязыватьСтроку(<НомерСтроки>)

Англоязычный синоним:

LinkLine

Параметры:

<НомерСтроки>	Выражение, содержащее номер строки документа.

Описание:

Метод ПривязыватьСтроку устанавливает, что при выполнении всех последующих движений регистров, при записи значений периодических реквизитов справочников с привязкой к документу, а также при записи бухгалтерских проводок движения будут записываться с номером строки <НомерСтроки>.

Метод ПривязыватьСтроку может быть использован как системная процедура: тогда он будет устанавливать номер строки для всех регистров по которым выполняются движения, или как метод конкретного регистра: тогда он будет устанавливать номер строки только для данного регистра.

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения.

Пример:

```
Процедура ОбработкаПроведения()
   ПривязыватьСтроку(НомерСтроки);
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока(ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

Предопределенные процедуры Модуля документа

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях документов (см. «Виды программных модулей»).

Данные предопределенные процедуры вызываются как при интерактивном так и при программном возникновении события.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ОбработкаПроведения

Предопределенная процедура обработки проведения документа.

Синтаксис:

```
ОбработкаПроведения(<Знач>)
```

Англоязычный синоним:

```
Posting
```

Параметры:

<Знач>	Необязательный параметр. Идентификатор переменной, в которую системой будет передано значение параметра, если проведение документа запущено программно при помощи метода Провести. Использовать данное значение можно, например, для того, чтобы в процедуре ОбработкаПроведения правильно отработать режим проведения, т. к. в случае вызова метода Провести это будет программный, а не интерактивный и не групповой (см. ГрупповаяОбработка) способ проведения документа.

Описание:

Процедура ОбработкаПроведения — предопределенная процедура. Она не является встроенной процедурой языка. Для нее определено только название и синтаксис. Тело процедуры должно быть написано разработчиком конфигурации в Модуле документа (см. «Виды программных модулей»). Собственно говоря, весь алгоритм проведения документа и заключается в данной процедуре.

Вызов процедуры ОбработкаПроведения производится в системе 1С:Предприятие неявно при выполнении проведения документа в любом режиме (интерактивное проведение документа, групповое проведение, программная инициализация проведения при помощи метода Провести). Если проведение документа запущено программно, то система подставляет фактическое значение параметра <Знач>, содержащее значение параметра, заданного при вызове метода Провести.

Замечание. В предопределенных процедурах ОбработкаПроведения и ОбработкаУдаленияПроведения система 1С:Предприятие сама выполняет все действия через транзакцию (см. «Процедуры работы с транзакциями»), поэтому никаких специальных действий по обработке транзакций в этих предопределенных процедурах предпринимать не нужно.

Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя проводить документ), то проведение документа не будет выполнено и будет отменена системная транзакция.

Замечание. В предопределенных процедурах ОбработкаПроведения и ОбработкаУдаленияПроведения нельзя использовать элементы интерактивного управления (например, операторы Предупреждение, Вопрос, ВвестиЧисло и т. п.), т. к. в этом случае при открытой транзакции система ожидает отклика пользователя, а все остальные пользователи ждут завершения транзакции (в результате документы у всех остальных пользователей в этот момент не могут проводиться). Если в конфигурации необходимо при проведении документа выдавать пользователю некоторые coofi щения, то следует использовать операторы Сообщить или Состояние

Для корректного выполнения алгоритма проведения документа в Модуле документа доступны методы ИтогиАктуальны и ГрупповаяОбработка, которые позволяют правильно определить режим проведения документа (см. «Работа с документами»).

Данная предопределенная процедура может располагаться только в Модулях документов (см. «Виды программных модулей»).

Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
```

```
    Пока( ПолучитьСтроку() > 0) Цикл
        Регистр.ТовЗап.Товар = Товар;
        Регистр.ТовЗап.Количество = Количество;
        Регистр.ТовЗап.ДвижениеРасходВыполнить();
    КонецЦикла;
КонецПроцедуры
```

См. также: СтатусВозврата, МтогиАктуальны, ГрупповаяОбработка, Провести

ОбработкаУдаленияПроведения

Предопределенная процедура обработки удаления проведения документа.

Синтаксис:

```
ОбработкаУдаленияПроведения()
```

Англоязычный синоним:

```
UnPostingProcess
```

Описание:

Процедура ОбработкаУдаленияПроведения — предопределенная процедура. Она не является встроенной процедурой языка. Для нее определено только название и синтаксис. Тело процедуры должно быть написано разработчиком конфигурации в Модуле документа (см. «Виды программных модулей»). Вызов процедуры ОбработкаУдаленияПроведения производится в системе 1С:Предприятие неявно при отмене проведения документа или при удалении проведенных документов в любом режиме — интерактивно или программно. (см. СделатьНеПроведенным, Удалить).

Замечание. В предопределенных процедурах ОбработкаПроведения и ОбработкаУдаленияПроведения система 1С:Предприятие сама выполняет все действия через транзакцию (см. «Процедуры работы с транзакциями»), поэтому никаких специальных действий по обработке транзакций в этих предопределенных процедурах предпринимать не нужно.

Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя отменять проведение документа) то отмена проведения документа не будет выполнена и будет отменена системная транзакция.

Замечание. В предопределенных процедурах ОбработкаУдаленияПроведения и ОбработкаПроведения нельзя использовать элементы интерактивного управления (например, операторы Предупреждение, Вопрос, ВвестиЧисло и т. п.), т. к. в этом случае при открытой транзакции система ожидает отклика пользователя, а все остальные пользователи ждут завершения транзакции (в результате документы у всех остальных пользователей в этот момент не могут проводиться). Если в конфигурации необходимо при проведении документа выдавать пользователю некоторые сообщения, то следует использовать операторы Сообщить или Состояние.

Данная предопределенная процедура может располагаться только в Модулях документов (см. «Виды программных модулей»).

Пример:

```
Процедура ОбработкаУдаленияПроведения()
    // Процедура выполняется при отмене проведения
    // или удалении документа
    // Сотрудник — это реквизит документа
    СпрСотрудники.НайтиЭлемент(Сотрудник);
    // если нашли сотрудника...
    Если СпрСотрудники.Выбран() = 1 Тогда
        // СтРасч — это реквизит документа
        СпрСотрудники.НайтиЭлемент(СтРасч);
        ВозвратРасч = 0;
        Если СпрСотрудники.Выбран() = 1 Тогда
            // если старый расчетчик еще есть — откатим!
            ВозвратРасч = 1;
        КонецЕсли;
        СпрСотрудники.НайтиЭлемент(Сотрудник);
        Если ВозвратРасч = 1 Тогда
            СпрСотрудники.Родитель = СтРасч;
        КонецЕсли;
        СпрСотрудники.Записать();
    КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, СделатьНеПроведенным, Удалить

АрхивироватьДокумент

Предопределенная процедура обработки архивирования документа.

Синтаксис:

```
АрхивироватьДокумент()
```

Англоязычный синоним:

```
ArchiveDocument
```

Описание:

Вызов процедуры АрхивироватьДокумент производится в системе 1С:Предприятие неявно в процессе смены расчетного периода журнала расчетов "вперед" (т. н. архивация данных расчета), если в журнале расчетов содержатся записи, порожденные данным документом.

Тело процедуры должно быть написано разработчиком конфигурации в Модуле документа, вызываемом в конфигураторе: Документ — Редактировать — Модуль документа.

Пример:

```
Процедура АрхивироватьДокумент()
   // Процедура выполняется при архивации документа расчета
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   Если Окончание > ЖР.КонецТекущегоПериода() Тогда
      Зарегистрировать(ТекущийДокумент());
   КонецЕсли;
КонецПроцедуры
```

Команды организации механизма заполнения документа методом подбора

При вводе документа, процесс заполнения реквизитов многострочной части документа может быть реализован как обычным способом, так и при помощи специального механизма — методом подбора. Этот механизм предназначен для ускорения интерактивного заполнения значений реквизитов документов типа «справочник» и «документ». Реальным примером может служить процесс заполнения спецификации накладной путем подбора нужных товаров из справочника "Товары".

В системе 1С:Предприятие для реализации подбора предназначены следующие системные процедуры:

- ОткрытьПодбор;
- Активизировать;
- АктивизироватьСтроку.

Кроме того, при обработке выбранных в подборе значений участвует следующая системная предопределенная процедура:

- ОбработкаПодбора.

Глава 13
Работа с Журналами документов

Контекст работы с журналами документов

Журнал документов — средство для работы со списком документов. В терминах языка журнал не является специальным типом данных (он не имеет значения, его нельзя создать при помощи функции `СоздатьОбъект`).

С журналом в системе связана форма отображения списка документов и программный модуль формы журнала документов (см. «Виды программных модулей»). В локальном контексте этого программного модуля непосредственно доступны реквизиты формы журнала. Кроме того, здесь непосредственно доступен атрибут «только для чтения» — `ТекущийДокумент`, содержащий значение выбранного в журнале документа.

Атрибуты контекста Модуля формы журнала документов

Описанный в данном разделе атрибут можно вызывать только в Модуле формы журнала (см. «Виды программных модулей»).

ТекущийДокумент

Синтаксис:
`ТекущийДокумент`
Англоязычный синоним:
`CurrentDocument`
Описание:
Атрибут (только для чтения) `ТекущийДокумент` содержит значение выбранного в журнале документа. Этот атрибут позволяет использовать в программе полученное значение документа. Поскольку данный атрибут фактически является ссылкой на позиционируемый объект типа «документ», то поэтому для передачи или запоминания самого значения текущего документа журнала рекомендуется использовать конструкцию
`ТекущийДокумент.ТекущийДокумент()`
Данный атрибут может использоваться только в локальном контексте программного модуля формы журнала документов.
Пример:

```
// только в модуле Формы журнала документа
// ————————————————————————————————
функция Операция()
   Если Число(ТекущийДокумент.Выбран()) > 0 Тогда
      Возврат "" + ТекущийДокумент.ДатаДок + " " + ТекущийДокумент.Вид;
   Иначе
      Возврат "Пусто";
   КонецЕсли;
КонецФункции
//————————————————————————————————
Функция ЖурналСумма()
   Если Число(ТекущийДокумент.Выбран()) > 0 тогда
      Возврат Константа.МинЗарплата.Получить(ДатаДок);
   Иначе
      Возврат 0;
   КонецЕсли;
КонецФункции
```

Методы контекста Модуля формы журнала документов

Описанные в данном разделе методы доступны только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Отбор по значениям может производиться только в «Общем» журнале, т. е. в котором установлен признак «Все документы». Таким образом, отбор по значениям может выступать в качестве альтернативного способа выбора документов в журнале, относительно выбору документов по видам. С другой стороны, сам набор ссылок на реквизиты документов определенных видов в настройке графы отбора определяет состав видов документов включаемых в данный отбор. Для того, чтобы при открытии журнала не показывался список всех документов, следует в предопределенной процедуре `ПриОткрытии` установить отбор по несуществующему значению отбора, например, по пустому значению.

ВидыОтбора

Установить доступные виды отборов журнала для вызова их в интерактивном режиме.

Синтаксис:

```
ВидыОтбора(<СписокИменОтборов>)
```

Англоязычный синоним:

```
KindsOfSelection
```

Параметры:

<СписокИменОтборов>	Необязательный параметр. Строковое выражение, содержащее список имен отборов для журнала. Виды указываются через запятую. Если вместо списка — символ "*", то значит для журнала используются все назначенные в конфигураторе виды отборов.

Возвращаемое значение:

Строковое значение, содержащее текущий список имен отборов для журнала, разделенных запятыми (на момент до исполнения метода).

Описание:

Метод ВидыОтбора устанавливает доступные виды отборов журнала для вызова их в интерактивном режиме.

Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Пример:

```
ВидыОтбора("Склад, Клиент, Автор");
```

ЗакладкиОтбора

Установить в форме журнала закладки для интерактивного осуществления отбора.

Синтаксис:

```
ЗакладкиОтбора(<ИмяОтбора>, <ФлагОтбораЗначенийВИнтервалеЖурнала>,
               <ФлагУстановкиЗначенияОтбора>, <Значение0тбора>)
```

Англоязычный синоним:

```
TabCtrlSelection
```

Параметры:

<ИмяОтбора>	Строковое выражение, содержащее имя отбора.
<ФлагОтбораЗначенийВИнтервалеЖурнала>	Числовое выражение: 1 — текущие значения отбора для закладок выбираются только по документам в установленном интервале журнала; 0 — текущие значения отбора для закладок выбираются по всем документам журнала.
<ФлагУстановкиЗначенияОтбора>	Числовое выражение: 1 — для отображения сразу выбирается отбор по параметру <Значение0тбора>; 0 — текущее значение отображаемой закладки отбора устанавливается на первое существующее значение.
<Значение0тбора>	Значение отбора.

Описание:

Метод ЗакладкиОтбора устанавливает в форме журнала закладки для интерактивного осуществления отбора.

Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Пример:

```
ЗакладкиОтбора("Склады", 1, 1, Константа.ОснСклад);
```

УстановитьОтбор

Установить отбор журнала.

Синтаксис:

```
УстановитьОтбор(<ИмяОтбора>, <Значение0тбора>)
```

Англоязычный синоним:

```
SetSelection
```

Параметры:

<ИмяОтбора>	Строковое выражение — имя отбора. Если это значение пустое, то отбор отключается.
<ЗначениеОтбора>	Значение отбора.

Описание:

Метод УстановитьОтбор принудительно устанавливает отбор для журнала.

Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Замечание. Во всех журналах, кроме журнала подчиненных документов, работает отбор по виду документа. В этом случае синтаксис вызова метода следующий:

```
УстановитьОтбор(<ВедДокумента>)
```

Параметры:

<ВидДокумента>	Строковое выражение — вид документа отбора.

Пример:

```
Изм = УстановитьОтбор("Склады", Склад1);
```

ПолучитьОтбор

Возвратить текущее значение отбора журнала.

Синтаксис:

```
ПолучитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)
```

Англоязычный синоним:

```
GetSelection
```

Параметры:

<ИмяОтбора>	Имя переменной, куда вернется строковое значение имени отбора.
<ЗначениеОтбора>	Имя переменной, куда вернется значение отбора.

Возвращаемое значение:

Числовое значение: 1 — есть отбор; 0 — если нет отбора.

Описание:

Метод ПолучитьОтбор возвращает текущее значение отбора журнала. Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Пример:

```
Изм = ПолучитьОтбор();
```

УстановитьИнтервал

Установить интервал журнала.

Синтаксис:

```
УстановитьИнтервал(<ДатаНач>, <ДатаКон>, <ФлагИзменения>)
```

Англоязычный синоним:

```
SetRange
```

Параметры:

<ДатаНач>	Выражение типа «дата» — начало интервала журнала.
<ДатаКон>	Выражение типа «дата» — конец интервала журнала.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения интервала журнала. 1 — пользователь может изменить интервал журнала интерактивно, 0 — пользователь не может интерактивно изменить интервал журнала.

Описание:

Метод УстановитьМнтервал позволяет установить интервал журнала. Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Пример:

```
УстановитьИнтервал(НИ, КИ);
```

НачалоИнтервала

Возвратить дату начала интервала журнала.

Синтаксис:

```
НачалоИнтервала()
```

Англоязычный синоним:

```
BegOfRange
```

Возвращаемое значение:

Значение даты начала интервала журнала.

Описание:

Метод НачалоИнтервала позволяет получить дату начала интервала журнала.
Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).

Пример:

```
НИ = НачалоИнтервала();
```

КонецИнтервала

Возвратить дату конца интервала журнала.

Синтаксис:

```
КонецИнтервала()
```

Англоязычный синоним:

```
EndOfRange
```

Возвращаемое значение:

Значение даты конца интервала журнала.

Описание:

Метод КонецИнтервала позволяет получить дату конца интервала журнала.
Данный метод доступен только в контексте Модуля формы журнала документов (см. «Виды программных модулей»).
Пример:
```
КИ = КонецИнтервала();
```

ПодчинениеДокументу

Установить документ-владелец для журнала подчиненных документов.
Синтаксис:
```
ПодчинениеДокументу([<Докум>], [<ФлагАвтоСменыПодчинен>])
```
Англоязычный синоним:
```
ParentDoc
```
Параметры:

<Докум>	Необязательный параметр. Выражение типа «документ» — значение документа, подчиненные документы к которому должен отображать журнал.
<ФлагАвтоСменыПодчинен>	Необязательный параметр. Числовое выражение: 1 — устанавливает способность автоматически обновлять отображение журнала при смене документа-владельца, если такой режим установлен в параметрах системы («Сервис» - «Один журнал»); 0 — снятие режима автоматического обновления отображения журнала.

Возвращаемое значение:
Текущее значение документа, которому подчинен журнал.
Описание:
Метод ПодчинениеДокументу позволяет установить документ-владелец для журнала подчиненных документов.
Данный метод доступен только в контексте Модуля формы журнала подчиненных документов (см. «Виды программных модулей»).
Пример:
```
ПодчинениеДокументу(Док, 1);
```

Предопределенные процедуры Модуля формы журнала документов

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях формы журналов документов (см. «Виды программных модулей»).
В основном данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.
Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ПриУстановкеОтбора

Предопределенная процедура при установке отбора.
Синтаксис:
```
ПриУстановкеОтбора(<ИмяРеквизОтбора>, <Значение>)
```
Англоязычный синоним:
```
OnSetSelectInJournal
```
Параметры:

<ИмяРеквизОтбора>	Строковое значение — название общего реквизита документа (как оно задано в конфигураторе), по которому производится отбор (например, "Автор").
<Значение>	Значение реквизита отбора. Допустим, у документов существует общий реквизит "Автор", по которому решено провести отбор, значит в этом параметре будет передано конкретное значение этого реквизита, по которому решено провести отбор (например, "Сидоров И. А.").

Описание:
Вызов предопределенной процедуры ПриУстановкеОтбора производится системой 1С:Предприятие неявно при интерактивной попытке установить отбор документов в журнале. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя устанавливать данное значение отбора документов), установка не будет выполнена.
Данная предопределенная процедура может располагаться в модуле формы журнала и в глобальном программном модуле. Если данная процедура присутствует в модуле формы соответствующего журнала, то будет вызвана она, если нет, то будет вызвана процедура из глобального модуля.

Пример:

```
Процедура ПриУстановкеОтбора(ИмяОтбора, ЗначОтбора) Если
   НазваниеНабораПрав() = "Продавец" Тогда
      Если (Имя0тбора = "Автор") И (ЗначОтбора <> ТекущПользователь) Тогда
         Предупреждение("У вас нет права просматривать чужие документы!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУстановкеИнтервала

Предопределенная процедура при установке интервала журнала.

Синтаксис:

```
ПриУстановкеИнтервала(<ДатаНач>, <ДатаКон>)
```

Англоязычный синоним:

```
OnSetRange
```

Параметры:

<ДатаНач>	Дата начала интервала журнала.
<ДатаКон>	Дата конца интервала журнала.

Описание:

Вызов предопределенной процедуры ПриУстановкеИнтервала производится системой 1С:Предприятие неявно при интерактивной попытке установить интервал в журнале документов. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя изменять интервал журнала), установка не будет выполнена.

Пример:

```
Процедура ПриУстановкеИнтервала(ДатаНач, ДатаКон) Если
   НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права изменять интервал журнала! , 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 14
Работа с Регистрами оперативного учета

Регистры — это специфический инструмент системы 1С:Предприятие, средство накопления сводной информации. Регистры являются внутренним инструментом конфигурации, к которым нет непосредственного доступа через интерфейс пользователя. Информация о результатах хозяйственных операций, которая появляется при оформлении документов, накапливается в регистрах при «проведении» документов, а отображение информации, содержащейся в регистрах, осуществляется при помощи формирования отчетов. Информация из регистров используется для анализа хозяйственной деятельности за прошедший период.

При создании регистра определяется, как будет обрабатываться, группироваться и сохраняться сводная информация. Регистры могут быть двух видов: оборотные регистры и регистры остатков.

Регистр остатков — это объект, предназначенный для ведения остатков «ресурсов» на момент времени с привязкой к некоторому набору «измерений».

Оборотный регистр — это объект, предназначенный для подсчета оборота «ресурсов» за заданный интервал времени с привязкой к некоторому набору «измерений».

Ресурсами регистра могут являться любые категории учета, которые могут быть представлены в числовом виде, такие как: «количество товара», «долг клиента», «сумма наличных» и т. п. Измерения регистра — это оси координат, на пересечении которых регистр хранит конкретные значения ресурсов. Кроме того, при записи движения регистра можно задавать «реквизиты» регистра. Значения реквизитов регистра просто сопровождают запись о движении регистра (подобно комментарию) для возможности в дальнейшем производить фильтрацию движений при выборке.

Пример:

* Предположим, что в конфигураторе создан регистр остатков «Товарные_Запасы», который должен содержать сведения об остатке каждого товара на каждом складе, за каждым материально ответственным лицом. В дальнейшем предполагается получать информацию такого типа: «остаток конкретного товара на конкретном складе», «остаток конкретного товара всего» и т. п. В идеологии системы 1С:Предприятие такой регистр представляет собой прямоугольную систему координат, измерениями этого регистра являются: «Товар», «Склад», «МОЛ». Регистр имеет один ресурс — «Количество». Содержание данного регистра можно представить в виде следующей таблицы, где по каждому конкретному товару на конкретном складе за конкретным МОЛ числится определенное Количество.

Товар	Склад	МОЛ	Количество
Стол	Первый	Иванов	10
Стол	Первый	Петров	20
Стол	Второй	Иванов	5
Шкаф	Первый	Петров	7
Шкаф	Второй	Иванов	2
Шкаф	Второй	Петров	8
Шкаф	Третий	Петров	10

Структура каждого конкретного вида регистра определяется при его создании в конфигураторе. Измерения, ресурсы и реквизиты регистров определяются в конфигураторе конкретно для каждого создаваемого вида регистра.

В отличие от документов и справочников, которые представлены конкретными объектами в информационной базе, регистр является механизмом для работы с итогами. Вследствие этого атрибуты регистра не являются свойствами некоторого объекта, а используются лишь как служебные переменные для манипуляции измерениями, ресурсами и реквизитами регистра при помощи соответствующих методов. Для каждого метода определяется, каким образом он использует атрибуты регистров. Следует обратить внимание, что ресурсы регистров принимают различное смысловое значение в зависимости от конкретной операции, выполняемой с регистром. Например, при записи движения регистра при помощи методов: `ДвижениеПриходВыполнить`, Движение `РасходВыполнить` и т. п. в ресурс регистра следует записывать приращение, которое будет изменять содержимое регистра, а при считывании итогов регистра (`Остатки`, `ОстаткиПолучить` и т. п.) ресурсы регистра содержат остаток (итоговые значения) содержимого регистров.

Под термином «остаток» ресурса для регистра остатков следует понимать числовое значение, которое имеет ресурс на какой-то момент времени. Именно здесь и проявляется понятие Точки Актуальности (далее ТА). ТА — это тот момент времени (дата+время), на который все необоротные регистры содержат текущие значения своих ресурсов. Другими словами, если просто запросить у регистра остатков значение какого-либо ресурса, то он выдаст его именно на момент ТА. Если необходимо узнать значение ресурса на какой-либо прошедший момент времени, то надо либо сдвинуть в прошлое ТА (для этого используется интерактивная операция «Управления итогами»), либо при помощи средств встроенного языка следует запустить «временный расчет» регистра. При «временном расчете» регистра остатков можно задать момент времени, на который необходимо получить значение ресурса. В этом случае значение ресурса рассчитывается, что требует несколько большего времени процессора, чем если получать ресурс на ТА. Однако, «временный расчет» рассчитывается не с начала всех времен, а от начала месяца. Регистры остатков хранят промежуточные значения ресурсов на начало каждого месяца (именно с этим связана процедура «переход на новый период»). Таким образом, если при «временном рас-

чете» указать момент времени в начале месяца, то, в общем случае, расчет будет выполнен быстрее, чем если указать конец месяца.

Кроме хранения остатка или оборота ресурсов, регистры хранят все «приращения» (приход со знаком «+», расход со знаком «-»), произведенные документами (это обозначается термином «движения регистра»). Причем, эти «при-оащения» регистров хранятся со ссылками на те документы, которые их вызвали. Поэтому, во встроенном языке системы 1С:Предприятие есть методы, чтобы получить из регистра все его движения (с привязкой к документам), указав интервал времени, за который они нужны.

Документы, как объекты прикладной задачи, тесно связаны с регистрами. Именно документы, и только они могут изменять значения ресурсов в регистрах (этот алгоритм прописывается в предопределенной процедуре `ОбработкаПроведения` для каждого вида документа). Никаким другим способом, кроме как через проведение документов на содержимое регистров повлиять нельзя. Например, нельзя создать отчет, который бы изменял значения, хранящиеся в регистрах. Движения (приращения) регистров выполняются в момент «проведения» документа. С другой стороны, документы не порождают в регистрах необратимых действий. Что это значит? Если ранее проведенный документ удалить или сделать его непроведенным, то удалятся и все движения регистров, порожденные им. Если отредактировать (изменить) ранее проведенный документ, то после перепроведения документа все движения регистров, порожденные ранее, удалятся и заменятся новыми.

Замечание: Формирование движений документов по регистрам доступно только в Модуле документа в системной предопределенной процедуре `ОбработкаПроведения`.

Контекст работы с регистрами

В синтаксисе языка применение атрибутов и вызов методов регистров может выполняться двумя способами:

- Средства языка предоставляют возможность непосредственного доступа к атрибутам и методам регистров, объявленных в конфигураторе в любом программном модуле (все объявленные в конфигураторе регистры *принадлежат глобальному контексту* конфигурации). Во всех текстах программных модулей доступ к атрибутам, вызовы методов регистров можно записывать просто через точку после полного имени регистра.

Пример:

```
Флаг = Регистр.ОстаткиТовара.Приход;
```

- Во всех программных модулях доступ к атрибутам и методам регистров __ может выполняться при помощи переменной со ссылкой на объект типа регистр. Объект создается функцией СоздатьОбъект, ссылка на который присваивается переменной. Чтобы вызвать атрибут или метод объекта, имя этого атрибута или метода (с указанием необходимых параметров) пишется через точку после имени объекта При создании объекта типа регистр, в качестве названия регистра обязательно должно выступать полное имя конкретного регистра, как оно объявлено в конфигураторе. Вид регистра записывается через точку после ключевого слова Регистр, т. е. полное имя регистра записывается следующим образом:

`Регистр.<XXXXX>`

где <XXXXX> — вид регистра, объявленный в конфигураторе. Англоязычный синоним ключевого слова `Регистр` — `Register`.

Пример:

```
Рег = СоздатьОбъект("Регистр.ТовЗап");
```

Атрибуты регистров

Приход

Флаг движения регистра «приход».

Синтаксис:

`Приход`

Англоязычный синоним:

`Income`

Описание:

Атрибут `Приход` содержит значение флага записи движения регистра и используется только при обращении к существующим записям движений регистров. Данный атрибут только для чтения. Атрибут `Приход` может принимать числовое значение 1 — если текущее движение регистра приход, 0 — если нет.

Данный атрибут не имеет смысла для оборотных регистров.

Пример:

```
Рег1 = СоздатьОбъект("Регистр.Взаиморасчеты");
Если Режим = "Подробно" Тогда
   Рег1.УстановитьФильтр(Клиент);
```

```
   Рег1.ВыбратьДвижения(ДатаНачало, );
   Пока Рег1.ПолучитьДвижение()>0 Цикл
      Если Число(Рег1.Клиент.Код) <> Число(Запрос.Клиент.Код) Тогда
         Продолжить;
      КонецЕсли;
      Док = Рег1.ТекущийДокумент();
      Дв_Баз_Сум = Рег1.СуммаБазовая;
      Дв_Вал_Сум = Рег1.СуммаВалютная;
      Если Рег1.Приход = 1 Тогда
         Таб.ВывестиСекцию("Приход");
      Иначе
         Таб.ВывестиСекцию("Расход");
      КонецЕсли;
   КонецЦикла;
КонецЕсли;
```

Расход

Флаг движения регистра «расход».

Синтаксис:

`Расход`

Англоязычный синоним:

`Outcome`

Описание:

Атрибут `Расход` содержит значение флага записи движения регистра и используется только при обращении к существующим записям движений регистров. Данный атрибут только для чтения. Атрибут `Расход` может принимать числовое значение 1 — если текущее движение регистра расход, 0 — если нет.

Данный атрибут не имеет смысла для оборотных регистров.

Пример:

```
Рег1 = СоздатьОбъект("Регистр.Взаиморасчеты");
Если Режим = "Подробно" Тогда
   Рег1.УстановитьФильтр(Клиент);
   Рег1.ВыбратьДвижения(ДатаНачало, );
   Пока Рег1.ПолучитьДвижение() > 0 Цикл
      Если Число(Рег1.Клиент.Код) <> Число(Запрос.Клиент.Код) Тогда
         Продолжить;
      КонецЕсли;
      Док = Рег1.ТекущийДокумент();
      Дв_Баз_Сум = Рег1.СуммаБазовая;
      Дв_Вал_Сум = Рег1.СуммаВалютная;
      Если Рег1.Расход = 1 Тогда
         Таб.ВывестиСекцию("Расход");
      Иначе
         Таб.ВывестиСекцию("Приход");
      КонецЕсли;
   КонецЦикла;
КонецЕсли;
```

<Измерение>

Установить значение измерения регистра.

Синтаксис:

`<Измерение>`

Описание:

Атрибут **<**`Измерение`**>** задает значение выбранного измерения регистра. В тексте программного модуля используется название конкретного измерения регистра, как он назван в конфигураторе.

Пример:

```
Процедура ОбработкаПроведения()
   // В регистре "ТовЗап" измерениями являются "Склад" и "Товар"
   // ресурсами являются "Количество" и "Стоимость"
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока (ПолучитьСтроку() > 0) Цикл
```

```
      Регистр.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

<Ресурс>

Установить значение ресурса регистра.

Синтаксис:

```
<Ресурс>
```

Описание:

Атрибут <Ресурс> задает значение выбранного ресурса регистра. В тексте программного модуля используется название конкретного ресурса регистра, как он назван в конфигураторе.

Пример:

См. предыдущий пример.

<Реквизит>

Установить значение реквизита регистра.

Синтаксис:

```
<Реквизит>
```

Описание:

Атрибут <Реквизит> задает значение выбранного реквизита регистра. В тексте программного модуля используется название конкретного реквизита регистра, как он назван в конфигураторе.

Пример:

```
Процедура ОбработкаПроведения()
   // В регистре "ТовЗап" измерениями являются "Склад" и "Товар"
   // ресурсами являются "Количество" и "Стоимость"
   // реквизитом является "ВнутрПеремещение"
   Регистр.ТовЗап.Склад = Склад;
   Если Вид() = "Перемещение" Тогда
      Регистр.ТовЗап.ВнутрПеремещение = 1;
   Иначе
      Регистр.ТовЗап.ВнутрПерсмещение = 0;
   КонецЕсли;
   ВыбратьСтроки();
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ТовЗап.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

Методы оборотных регистров

Описанные в данном разделе методы используются только для оборотных регистров.

ИспользоватьПериод

Установить период выборки итогов оборотного регистра.

Синтаксис 1:

```
ИспользоватьПериод(<Дата>)
```

Синтаксис 2:

```
ИспользоватьПериод(<Год>, <Месяц>, <День>)
```

Синтаксис 3:

```
ИспользоватьПериод(<Год>, <НомерНедели>)
```

Синтаксис 4:

```
ИспользоватьПериод (<Год>, <Месяц>, <НомерДекады>)
```

Синтаксис 5:

ИспользоватьПериод(<Год>, <Месяц>)

Синтаксис 6:

ИспользоватьПериод(<Год>, <НомерКвартала>)

Синтаксис 7:

ИспользоватьПериод(<Год>)

Англоязычный синоним:

UsePeriod

Параметры:

<Дата>	Выражение типа «дата».
<Год>	Числовое выражение, содержащее номера года.
<Месяц>	Числовое выражение, содержащее номер месяца.
<День>	Числовое выражение, содержащее номер дня месяца.
<НомерДекады>	Числовое выражение, содержащее номер декады месяца.
<НомерКвартала>	Числовое выражение, содержащее номер квартала.

Описание:

Метод ИспользоватьПериод устанавливает период выборки итогов оборотного регистра. Различный синтаксис вызова данного метода зависит от заданного в конфигураторе интервала оборотного регистра.

Заданный в конфигураторе Интервал	Используемый синтаксис
День	синтаксис 1 и 2.
Неделя	синтаксис 1 и 3.
Декада	синтаксис 4.
Месяц	синтаксис 5.
Квартал	синтаксис 6.
Год	синтаксис 7.

Данный метод используется до вызова методов оборотных регистров. Дальнейшая выборка итогов будет происходить по указанному периоду. Если при работе с оборотным регистром этот метод опущен, то по умолчанию используется текущий период, в котором находится ТА.

Пример:

```
Регистр.ОборотыТоваров.ИспользоватьПериод(1996);
```

Итог

Возвратить итог одного ресурса оборотного регистра.

Синтаксис:

Итог(<Измерен1>, <Измерен2>..., <ИмяРесурса>)

Англоязычный синоним:

Total

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.
<ИмяРесурса>	Строковое выражение, содержащее название требуемого ресурса регистра, как оно названо в конфигураторе.

Возвращаемое значение:

Численое значение итога заданного ресурса регистра.

Описание:

Метод Итог возвращает итог по ранее установленному периоду ресурса <ИмяРесурса> оборотного регистра.
Данный метод используется только для оборотных регистров.

Пример:

```
Функция ПолучитьИтог(Товар, Склад, Год);
   Регистр.ОборотыТоваров.ИспользоватьПериод(Год);
   Возврат Регистр.ОборотыТоваров.Итог(Товар, Склад, "ОборотТовара");
КонецФункции
```

Итоги

Извлечь все итоги ресурсов оборотного регистра.

Синтаксис:

Итоги(<Измерен1>, <Измерен2>...)

Англоязычный синоним:

Totals

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.

Описание:

Метод Итоги извлекает итоги по всем ресурсам по ранее установленному периоду оборотного регистра. Полученные итоги ресурсов затем можно считывать из атрибутов регистра.

Данный метод используется только для оборотного регистра.

Пример:

```
Регистр.ОборотыТоваров.ИспользоватьПериод(Год);
Регистр.ОборотыТоваров.Итоги(Товар, Склад);
ОбТов = Регистр.ОборотыТоваров.ОборотТовара;
```

СводныйИтог

Возвратить сводный итог ресурса оборотного регистра.

Синтаксис:

```
СводныйИтог(<Измерен1>, <Измерен2>..., <ИмяРесурса>)
```

Англоязычный синоним:

```
ConsolidatedTotal
```

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.
<ИмяРесурса>	Строковое выражение, содержащее название требуемого ресурса регистра, как оно названо в конфигураторе.

Возвращаемое значение:

Численое значение сводного итога заданного ресурса регистра.

Описание:

Метод СводныйИтог возвращает сводный итог ресурса <ИмяРесурса> оборотного регистра. Данный метод работает аналогично методу Итог, однако здесь могут быть заданы не все значения измерений, а только несколько, но обязательно в правильной последовательности, соответствующей структуре измерений данного регистра, как она задана в конфигураторе. Измерения могут задаваться с пропусками (неуказанное значение — просто запятая), фиксируются только указанные измерения.

Данный метод используется только для оборотного регистра.

Пример:

```
Функция ПолучитьСводныйИтог(Товар, Год);
   Регистр.ОборотыТоваров.ИспользоватьПериод(Год);
   Возврат Регистр.ОборотыТоваров.СводныйИтог(Товар, , "ОборотТовара");
КонецФункции
```

СводныеИтоги

Извлечь сводные итоги ресурсов оборотного регистра.

Синтаксис:

```
СводныеИтоги(<Измерен1>, <Измерен2>...)
```

Англоязычный синоним:

```
ConsolidatedTotals
```

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.

Описание:

Метод СводныеИтоги извлекает по переданным в качестве параметров измерениям сводные итоги всех ресурсов оборотного регистра. Полученные ресурсы затем можно считывать из атрибутов регистра. Метод работает аналогично процедуре Итоги, однако здесь могут быть заданы не все значения измерений, а только несколько, но обязательно в правильной последовательности, соответствующей структуре измерений данного регистра, как она задана в конфигураторе. Измерения могут задаваться с пропусками (неуказанное значение -просто запятая), фиксируются только указанные измерения.

Данный метод используется только для оборотного регистра.

Пример:

```
Регистр.ОборотыТоваров.ИспользоватьПериод(Год);
Регистр.ОборотыТоваров.СводниеИтоги(Товар, );
ПолныйОбТов = Регистр.ОборотыТоваров.ОборотТовара;
```

ИтогиПолучить

Получить все ресурсы по регистру.

Синтаксис:

```
ИтогиПолучить()
```

Англоязычный синоним:

TotalsGet

Описание:

Метод ИтогиПолучить извлекает все итоги ресурсов оборотного регистра по измерениям, содержащимся в атрибутах. Полученные ресурсы затем можно считывать из атрибутов регистра.

Данный метод используется только для оборотного регистра.

Пример:

```
Регистр.ОборотыТоваров.Товар = Конт.Товар;
Регистр.ОборотыТоваров.Склад = Конт.Склад;
Регистр.ОборотыТоваров.ИтогиПолучить();
Сообщить("Оборот товара=" + Регистр.ОборотыТоваров.ОборотТовара;
```

Методы регистров остатков

Описанные в данном разделе методы используются только для регистров остатков.

Остаток

Возвратить остаток одного ресурса регистра.

Синтаксис:

Остаток(<Измерен1>, <Измерен2>..., <ИмяРесурса>)

Англоязычный синоним:

Rest

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.
<ИмяРесурса>	Строковое выражение, содержащее название требуемого ресурса регистра, как оно задано в конфигураторе.

Возвращаемое значение:

Численое значение остатка заданного ресурса регистра.

Описание:

Метод Остаток возвращает остаток ресурса <ИмяРесурса> по регистру. Данный метод используется только для регистров остатков.

Пример:

```
Процедура ПолучитьОстаток(Товар, Склад, Ост, Стоим);
   Ост = Регистр.УчетТовара.Остаток(Товар, Склад, "ОстаткиТовара");
   Стоим = Регистр.УчетТовара.Остаток(Товар, Склад, "СтоимостьТовара");
КонецПроцедуры
```

СводныйОстаток

Возвратить сводный остаток ресурса по регистру

Синтаксис:

СводныйОстаток(<Измерен1>, <Измерен2>..., <ИмяРесурса>)

Англоязычный синоним:

ConsolidatedRest

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.
<ИмяРесурса>	Строковое выражение, содержащее название требуемого ресурса регистра, как оно названо в конфигураторе.

Возвращаемое значение:

Численное значение сводного остатка заданного ресурса регистра.

Описание:

Метод СводныйОстаток возвращает сводный остаток ресурса <ИмяРесурса> по регистру. Данный метод работает аналогично методу Остаток, однако здесь могут быть заданы не все значения измерений, а только несколько, но обязательно в правильной последовательности, соответствующей структуре измерений данного регистра, как она задана в конфигураторе. Измерения могут задаваться с пропусками (неуказанное значение — просто запятая), фиксируются только указанные измерения.

Данный метод используется только для регистров остатков.

Пример:

* Для регистра с измерениями «Товар», «Склад», «МОЛ» запись

```
Рег.СводныйОстаток(Тов1, , "ОстатокТовара");
```

означает, что ресурс «ОстатокТовара» будет получен только по одному товару (Тов1), но по всем складам и всем МОЛам.

*

```
Процедура ПолучитьСводныйОстаток(Товар, Ост);
   Ост = Регистр.УчетТовара.СводныйОстаток(Товар, , "ОстаткиТовара");
КонецПроцедуры
```

Остатки

Извлечь все ресурсы по регистру.

Синтаксис:

```
Остатки(<Измерен1>, <Измерен2>...)
```

Англоязычный синоним:

```
Rests
```

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.

Описание:

Метод `Остатки` извлекает по переданным в качестве параметров измерениям все ресурсы по регистру. Полученные ресурсы затем можно считывать из атрибутов регистра.

Данный метод используется только для регистров остатков.

Пример:

```
Процедура ПолучитьОстатки(Товар, Склад, ОстатокТовара, Стоимость);
   Регистр.УчетТовара.Остатки(Товар, Склад);
   ОстатокТовара = Регистр.УчетТовара.ОстаткиТовара;
   Стоимость = Регистр.УчетТовара.СтоимостьТовара;
КонецПроцедуры
```

СводныеОстатки

Извлечь сводные ресурсы по регистру.

Синтаксис:

```
СводныеОстатки(<Измерен1>, <Измерен2>...)
```

Англоязычный синоним:

```
ConsolidatedRests
```

Параметры:

<Измерен1>, <Измерен2>...	Выражения со значениями измерений регистра.

Описание:

Метод `СводныеОстатки` извлекает по переданным в качестве параметров измерениям все ресурсы по регистру. Полученные ресурсы затем можно считывать из атрибутов регистра. Метод работает аналогично процедуре `Остатки`, однако здесь могут быть заданы не все значения измерений, а только несколько, но обязательно в правильной последовательности, соответствующей структуре измерений данного регистра, как она задана в конфигураторе. Измерения могут задаваться с пропусками (неуказанное значение — просто запятая), фиксируются только указанные измерения.

Данный метод используется только для регистров остатков.

Пример:

* Для регистра с измерениями «Товар», «Склад», «МОЛ» запись

```
Рег.СводныеОстатки(Тов1, , );
```

означает, что ресурсы будут получены только по одному товару (Тов1), но по всем складам и всем МОЛам.

*

```
Процедура ПолучитьСводныеОстатки(Товар, ОстатокТовара, Стоимость);
   Регистр.УчетТовара.СводныеОстатки(Товар, );
   ОстатокТовара = Регистр.УчетТовара.ОстаткиТовара;
   Стоимость = Регистр.УчетТовара.СтоимостьТовара;
КонецПроцедуры
```

ОстаткиПолучить

Получить все ресурсы по регистру.

Синтаксис:

```
ОстаткиПолучить()
```

Англоязычный синоним:

```
GetRests
```

Описание:

Метод `ОстаткиПолучить` извлекает все ресурсы регистра по измерениям содержащимся в атрибутах. Полученные ресурсы затем можно считывать из атрибутов регистра.

Данный метод используется только для регистров остатков.

Пример:

```
Регистр.Взаиморасчеты.Клиент = Конт.Клиент;
```

```
Регистр.Взаиморасчеты.ОстаткиПолучить();
Сообщить("Долг клиента по заиморасчетам="
         + СокрЛ(Регистр.Взаиморасчеты.СуммаВалютная);
```

ВыбратьДвиженияСОстатками

Выбрать все движения регистра с остатками.

Синтаксис:

```
ВыбратьДвиженияСОстатками(<КонецВыборки>, <ГрафаОтбора>)
```

Англоязычный синоним:

```
SelectActsWithRests
```

Параметры:

<КонецВыборки>	Выражение типа дата, документ или позиция документа задающее конец временного интервала выбора движений регистра.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора. "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:

Метод ВыбратьДвиженияСОстатками инициирует выбор всех движении регистра в указанном интервале дат. До вызова данной процедуры может быть установлен фильтр (см. УстановитьФильтр), ограничивающий выборку значений из регистра. Если <КонецВыборки> не указан или равен 0, то конном временного интервала выбора движений регистра считается ТА.

Данный метод отличается от метода ВыбратьДвижения тем, что после получения очередного движения метод Остаток будет выдавать актуальные значения остатков для этого регистра. Но для этого перед вызовом метода ВыбратьДвиженияСОстатками регистру должен быть установлен признак временногоРасчета» (см. ВременныйРасчет) и должен быть выполнен временный расчет (см. РассчитатьРегистрыНа и РассчитатьРегистрыПо) — его дата и будет являться начальной датой выбираемых движений регистра. Данная возможность не может быть использована совместно с дальнейшими «Временными» расчетами.

Собственно выбор первого и последующих движений регистра осуществляется методом ПолучитьДвижение.

Данный метод используется только для регистров остатков.

Пример:

```
Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.ВременныйРасчет();
Рег.УстановитьФильтр(Клиент, );
РассчитатьРегистрыНа(ДатаНачала);
Рег.ВыбратьДвиженияСОстатками(ДатаКонца);
Пока Рег.ПолучитьДвижение() = 1 Цикл
   Прих = 0;
   Расх = 0;
   Если Рег.Приход = 1 Тогда
      Прих = Рег.Долг;
   Иначе
      Расх = Рег.Долг;
   КонецЕсли;
   ТекущееСальдо = Рег.Остаток(Клиент, "Задолженность");
   Док = Рег.ТекущийДокумент();
   Таб.ВывестиСекцию("Документ");
КонецЦикла;
```

См. также: УстановитьФильтр, ПолучитьДвижение, ВыбратьДвижения, ВременныйРасчет, РассчитатьРегистрыНа, РассчитатьРегистрыПо

Общие методы регистров

Описанные в данном разделе методы используются как для регистров остатков так и для оборотных регистров.

Вид

Определить название вида регистра.

Синтаксис:

```
Вид()
```

Англоязычный синоним:
Kind
Возвращаемое значение:
Строковое значение, содержащее название вида регистра.
Описание:
Метод Вид позволяет определить название вида регистра, как он задан ь конфигураторе.
Пример:
```
// отобразим вид регистра в строке состояния
Состояние(Рег1.Вид());
```

ПредставлениеВида

Определить пользовательское представление вида регистра.
Синтаксис:
ПредставлениеВида()
Англоязычный синоним:
KindPresent
Возвращаемое значение:
Строковое значение, содержащее пользовательское представление вида регистра (синоним регистра или, если он пустой, то идентификатор).
Описание:
Метод ПредставлениеВида позволяет получить пользовательское представление вида регистра, как оно задано в конфигураторе.
Пример:
```
// отобразим пользовательское представление в строке состояния Состояние
(Рег1.ПредставлениеВида());
```

НазначитьТип

Назначить тип для реквизита неопределенного вида.
Синтаксис:
НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>)
Англоязычный синоним:
SetType
Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита регистра неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту регистра. Например: "Строка", "Число", "Справочник.Товары", " Документ.РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных. Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:
Метод НазначитьТип позволяет назначить тип для реквизита, которому в конфигураторе назначен тип «Неопределенный».
Пример:
```
Регистр.УчетТМЦ.НазначитьТип("ТМЦ", "Справочник.Товары");
```

УстановитьАтрибут

Установить значение атрибута по имени идентификатора.
Синтаксис:
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
Англоязычный синоним:
SetAttrib
Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя атрибута, как оно задано в конфигураторе.
<Значение>	Выражение, содержащее устанавливаемое значение атрибута.

Описание:

Метод УстановитьАтрибут позволяет установить значение атрибута по имени идентификатора, как оно задано в конфигураторе.

Пример;

```
Рег.УстановитьАтрибут("Сумма", СуммаТов);
```

ПолучитьАтрибут

Получить значение атрибута по идентификатору.

Синтаксис:

ПолучитьАтрибут(<ИмяАтрибута>)

Англоязычный синоним:

GetAttrib

Параметры:

<ИмяАтрибута>	Строковое выражение, содержащее имя атрибута, как оно задано в конфигураторе.

Возвращаемое значение:

Значение атрибута <ИмяАтрибута>.

Описание:

Метод ПолучитьАтрибут позволяет получить значение атрибута по идентификатору, как оно задано в метаданных.

Пример:

```
СуммаТов = Рег.ПолучитьАтрибут("Сумма");
```

ВыбратьДвижения

Выбрать все движения регистра по датам.

Синтаксис:

ВыбратьДвижения(<НачалоВыборки>, <КонецВыборки>, <ГрафаОтбора>)

Англоязычный синоним:

SelectActs

Параметры:

<НачалоВыборки>	Выражение типа дата, документ или позиция документа, задающее начало временного интервала выбора движений регистра.
<КонецВыборки>	Выражение типа дата, документ или позиция документа задающее конец временного интервала выбора движений регистра.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора. "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:

Метод ВыбратьДвижения инициирует выбор всех движений регистра в указанном интервале дат. До вызова данной процедуры может быть установлен фильтр (см. УстановитьФильтр), ограничивающий выборку значений из регистра. Если <КонецВыборки> не указан или равен 0, то концом временного интервала выбора движений регистра считается ТА.

Собственно выбор первого и последующих движений регистра осуществляется функцией ПолучитьДвижение.

Пример:

```
Рег1 = СоздатьОбъект("Регистр.Взаиморасчеты");
Если Режим = "Подробно" Тогда
   Рег1.УстановитьФильтр(Клиент);
   Рег1.ВыбратьДвижения(ДатаНачало, );
   Пока Рег1.ПолучитьДвижение() > 0 Цикл
      Док = Рег1.ТекущийДокумент();
      Дв_Баз_Сум = Рег1.СуммаБазовая;
      Дв_Вал_Сум = Рег1.СуммаВалютная;
      Если Рег1.Расход = 1 Тогда
         Таб.ВывестиСекцию("Расход2");
      Иначе
         Таб.ВывестиСекцию("Приход2");
      КонецЕсли;
   КонецЦикла;
КонецЕсли;
```

См. также: ПолучитьДвижение

ВыбратьДвиженияДокумента

Выбрать все движения регистра по документу.

Синтаксис:

ВыбратьДвиженияДокумента(<Документ>)

Англоязычный синоним:

SelectDocActs

Параметры:

<Документ> Значение типа Документ.

Возвращаемое значение:

Число: 1 — если действие выполнено и в выборке есть хотя бы один элемент; 0 — если действие не выполнено или в выборке нет ни одного элемента.

Описание:

Метод ВыбратьДвиженияДокумента инициирует выбор всех движений регистра по указанному документу <Документ>.

Собственно выбор первого и последующих движений регистра осуществляется функцией ПолучитьДвижение.

Пример:

```
Процедура Взаиморасчеты(ВыбКлиент, ДатаНачало, ДатаКонец)
   // Создание Таблицы для выходного отчета
   Заг = "Неизвестно.";
   Таб = СоздатьОбъект("Таблица");
   Клн = СоздатьОбъект("Справочник.Клиенты");
   Клн.ИспользоватьДату(ДатаКонец);
   Док = СоздатьОбъект("Документ");
   Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
   Рег1 = СоздатьОбъект("Регистр.Взаиморасчеты");
   Рег.ВременныйРасчет();
   РассчитатьРегистрыНа(ДатаКонец);
   Если ВыбКлиент.Выбран() = 0 Тогда
      //Без условий
      Заг = "По всем клиентам.";
   ИначеЕсли ВыбКлиент.ЭтоГруппа() = 1 Тогда
      Клн.ВключатьПодчиненные(ВыбКлиент);
      Клн.ИспользоватьРодителя(ВыбКлиент);
      Заг = "По клиентам группы " + ВыбКлиент.Наименование;
   Иначе
      Предупреждение("Выберите группу клиентов!");
      Возврат;
   КонецЕсли;
   Таб.ВывестиСекцию("Отчет");
   Клн.ВыбратьЭлементы();
   Пока Клн.ПолучитьЭлемент() > 0 Цикл
      Если Клн.ЭтоГруппа() = 1 Тогда
         Таб.ВывестиСекцию("Группа");
      Иначе
         Рег.Клиент = Клн.ТекущийЭлемент();
         Рег.ОстаткиПолучить();
         Баз_сум = Рег.СуммаБазовая;
         Вал_сум = Рег.СуммаВалютная;
         Таб.ВывестиСекцию("Клиент");
         // цикл по всем документам
         Док.ВыбратьДокументы(ДатаНачало, ДатаКонец);
         Пока Док.ПолучитьДокумент() > 0 Цикл
            // отфильтруем документы
            Если (Док.Вид() = "Перемещение") ИЛИ (Док.Вид() = "Счет") ИЛИ
                 (Док.Вид() = "Списание") ИЛИ
                 (Док.Вид() = "РучноеИзмОстатка") Тогда
               // Эти Документы не изменяют Взаиморасчеты
               Продолжить;
            КонецЕсли;
            // выберем все движения регистра по документу
            Рег1.ВыбратьДвиженияДокумента(Док.ТекущийДокумент());
```

```
            Пока Рег1.ПолучитьДвижение() > 0 Цикл
               Если Строка(Рег1.Клиент.Код) <> Строка(Клн.Код) Тогда
                  // не тот клиент
                  Продолжить;
               КонецЕсли;
               Дв_Баз_Сум = Рег1.СуммаБазовая;
               Дв_Вал_Сум = Рег1.СуммаВалютная;
               Если Рег1.Приход = 1 Тогда
                  Таб.ВывестиСекцию("Приход");
               Иначе
                  Таб.ВывестиСекцию("Расход");
               КонецЕсли;
            КонецЦикла;
         КонецЦикла;
      КонецЕсли;
   КонецЦикла;
   //Вызов выходного отчета в окно просмотра и редактирования.
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 4, 0);
   Таб.Показать("Взаиморасчеты с клиентами", "");
   ВыбКлиент = 0;
КонецПроцедуры
```

См. также: ПолучитьДвижение

ОбратныйПорядок

Установить порядок выборки документов.

Синтаксис:

ОбратныйПорядок(<Режим>)

Англоязычный синоним:

BackwardOrder

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — выбирать движения документов в обратном порядке даты и времени; 0 — выбирать движения документов в порядке возрастания даты и времени.

Возвращаемое значение:

Числовое значение, содержащее текущее значение режима порядка выборки документов (на момент до исполнения метода).

Описание:

Метод ОбратныйПорядок устанавливает порядок выборки движения документов. Данный метод обычно используется до вызова одного из методов: ВыбратьДвижения, ВыбратьДвиженияСОстатками, который фактически открывает выборку. Дальнейшая выборка при помощи метода ПолучитьДвижение будет происходить в заданном порядке выборки.

По умолчанию, выборка движения документов выполняется в порядке возрастания даты и времени записи документов. Поэтому реально имеет смысл применять данный метод только в том случае, если надо получить обратный порядок выборки.

В тексте программного модуля возможно использование данного метода как процедуры или как функции. При использовании в качестве функции, возвращаемое значение соответствует текущему порядку выборки, которое было до вызова данного метода.

Пример:

```
Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
Если Режим = "Подробно" Тогда
   Рег1.УстановитьФильтр(Клиент);
   // Открываем выборку
   Рег1.ОбратныйПорядок(1);
   Рег1.ВыбратьДвижения(ДатаНачало, );
   // Цикл выбора движений по регистру
   Пока Рег1.ПолучитьДвижение() > 0 Цикл
      Если Строка(Рег1.Клиент.Код) <> Строка(Запрос.Клиент.Код) Тогда
         Продолжить;
      КонецЕсли;
      Док = Рег1.ТекущийДокумент();
```

```
      Дв_Баз_Сум = Рег1.СуммаБазовая;
      Дв_Вал_Сум = Рег1.СуммаВалютная;
      Если Рег1.Расход = 1 Тогда
         Таб.ВывестиСекцию("Расход2");
      Иначе
         Таб.ВывестиСекцию("Приход2");
      КонецЕсли;
   КонецЦикла;
КонецЕсли;
```

ПолучитьДвижение

Выбрать очередное движение регистра.

Синтаксис:

`ПолучитьДвижение()`

Англоязычный синоним:

`GetDocAct`

Возвращаемое значение:

Число: 1 — если следующее движение регистра выбрано успешно; 0 — если движение регистра не найдено (отсутствует).

Описание:

Метод `ПолучитьДвижение` выбирает следующее движение регистра в последовательности выборки, открытой перед этим при помощи методов `ВыбратьДвиженияДокумента` или `ВыбратьДвижения` или `ВыбратьДвиженияСОстатками`. Данный метод используется для организации цикла по регистру.

После выполнения даннго метода все атрибуты регистра: измерения, ресурсы, `Приход` и `Расход` содержат значения текущей записи движения регистра.

Данный метод может использоваться только для позиционируемых объектов, созданных функцией `СоздатьОбъект`.

Пример:

```
Рег1 = СоздатьОбъект("Регистр.Взаиморасчеты");
Если Режим = "Подробно" Тогда
   Рег1.УстановитьФильтр(Клиент);
   // Открываем выборку
   Рег1.ВыбратьДвижения(ДатаНачало, );
   // Цикл выбора движений по регистру
   Пока Рег1.ПолучитьДвижение() > 0 Цикл
      Если Строка(Рег1.Клиент.Код) <> Строка(Запрос.Клиент.Код) Тогда
         Продолжить;
      КонецЕсли;
      Док = Рег1.ТекущийДокумент();
      Дв_Баз_Сум = Рег1.СуммаБазовая;
      Дв_Вал_Сум = Рег1.СуммаВалютная;
      Если Рег1.Расход = 1 Тогда
         Таб.ВывестиСекцию("Расход2");
      Иначе
         Таб.ВывестиСекцию("Приход2");
      КонецЕсли;
   КонецЦикла;
КонецЕсли;
```

См. также: `ВыбратьДвиженияДокумента`, `ВыбратьДвижения`, `ВыбратьДвиженияСОстатками`

ТекущийДокумент

Возвратить значение документа, выполнившего движение регистра.

Синтаксис:

`ТекущийДокумент()`

Англоязычный синоним:

`CurrentDocument`

Возвращаемое значение:

Значение документа, который задал движение регистра.

Описание:

Метод `ТекущийДокумент` возвращает значение документа, который задал движение регистра. Метод используется после получения очередного движения регистра (см. методы `ПолучитьДвижение` и `ПолучитьИтог`).

Данный метод может использоваться только для переменных созданных функцией `СоздатьОбъект`.

Пример:
```
Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.ВременныйРасчет();
Рег.УстановитьФильтр(Клиент, );
РассчитатьРегистрыНа(ДатаНачала);
Рег.ВыбратьДвиженияСОстатками(ДатаКонца);
Пока Рег.ПолучитьДвижение() = 1 Цикл
   Прих = 0;
   Расх = 0;
   Если Рег.Приход = 1 Тогда
      Прих = Рег.Долг;
   Иначе
      Расх = Рег.Долг;
   КонецЕсли;
   ТекущееСальдо = Рег.Остаток(Клиент, "Задолженность");
   Док = Рег.ТекущийДокумент();
   Таб.ВывестиСекцию("Документ");
КонецЦикла;
```
См. также: ПолучитьДвижение, НомерСтроки

НомерСтроки

Возвратить номер строки документа, выполнившего движение регистра.

Синтаксис:
```
НомерСтроки()
```
Англоязычный синоним:
```
LineNum
```
Возвращаемое значение:

Число — номер строки документа, выполнившего движение регистра.

Описание:

Метод НомерСтроки возвращает значение номера строки документа, которая задала движение регистра (в случае, когда в Модуле документа перед движением регистра использовали метод ПривязыватьСтроку). Метод используется после получения очередного движения регистра (см. функции ПолучитьДвижение и ПолучитьИтог).

Данный метод может использоваться только для переменных созданных Функцией СоздатьОбъект.

Пример:
```
Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.ВременныйРасчет();
Рег.УстановитьФильтр(Клиент);
РассчитатьРегистрыНа(ДатаНачала);
Рег.ВыбратьДвиженияСОстатками(ДатаКонца);
Пока Рег.ПолучитьДвижение() = 1 Цикл
   Прих = 0;
   Расх = 0;
   Если Рег.Приход = 1 Тогда
      Прих = Рег.Долг;
   Иначе
      Расх = Рег.Долг;
   КонецЕсли;
   ТекущееСальдо = Рег.Остаток(Клиент, "Задолженность");
   Док = Рег.ТекущийДокумент();
   НомСтр = Рег.НомерСтроки();
   Таб.ВывестиСекцию("Документ");
КонецЦикла;
```
См. также: ПолучитьДвижение, ТекущийДокумент, ПривязыватьСтроку

ВыбратьИтоги

Выбрать все остатки регистра.

Синтаксис:
```
ВыбратьИтоги()
```
Англоязычный синоним:
```
SelectTotals
```
Описание:

Метод ВыбратьИтоги инициирует перебор текущих или временных (см. ВременныйРасчет) остатков по регистру, при этом используется фильтр, если он установлен (см. УстановитьФильтр). Открывается выборка остатков по всем записанным значениям измерений.

Собственно выбор первого и последующих остатков регистра осуществляется функцией ПолучитьИтог.

Пример:

```
Процедура ВыбратьКредит()
   Список = СоздатьОбъект("СписокЗначений");
   Рег = СоздатьОбъект("Регистр.ТоварныйКредит");
   Рег.УстановитьФильтр(Клиент, );
   Рег.ВыбратьИтоги();
   Ном = 0;
   Пока Рег.ПолучитьИтог() = 1 Цикл
      Док = Рег.Документ;
      Список.ДобавитьЗначение(Док, "" + Док + " — Остаток = " + Рег.Долг);
      Если Ном = 0 Тогда
         ВыбДок = Док;
      КонецЕсли;
      Ном = Ном + 1;
   КонецЦикла;
   Если Ном > 0 Тогда
      Если Список.ВыбратьЗначение(ВыбДок, "Выберите кредит") = 1 Тогда
         ДокКредита = ВыбДок;
      КонецЕсли;
   КонецЕсли;
   Возврат;
КонецПроцедуры
```

См. также: УстановитьФильтр, ПолучитьИтог, ВыбратьДвижения, ВременныйРасчет, РассчитатьРегистрыНа, РассчитатьРегистрыПо

ПолучитьИтог

Выбрать очередной остаток по регистру.

Синтаксис:

```
ПолучитьИтог()
```

Англоязычный синоним:

```
GetTotal
```

Возвращаемое значение:

Число: 1 — если очередной остаток по регистру выбран успешно; 0 — если очередной остаток по регистру не найден (отсутствует).

Описание:

Метод ПолучитьИтог выбирает очередной остаток по регистру в последовательности выборки, открытой перед этим при помощи метода ВыбратьИтоги. Данный метод используется для организации цикла по регистру — позволяет перебрать текущие или временные (см. ВременныйРасчет) остатки по регистру. При этом используется фильтр, если он установлен (см. УстановитьФильтр). Открывается выборка остатков по всем записанным значениям измерений.

Итоги с нулевыми остатками не выдаются. Порядок выдачи для измерений типа «справочник» и «документ» не определен.

После выполнения данного метода все атрибуты регистра: измерения, ресурсы, Приход и Расход содержат значения очередной записи остатков регистра. Данный метод может использоваться только для позиционируемых объектов, созданных функцией СоздатьОбъект.

Пример:

```
Процедура ВыбратьКредит()
   Список = СоздатьОбъект("СписокЗначений");
   Рег = СоздатьОбъект("Регистр.ТоварныйКредит");
   Рег.УстановитьФильтр(Клиент, );
   Ном = 0;
   Рег.ВыбратьИтоги();
   Пока Рег.ПолучитьИтог() = 1 Цикл
      Док = Рег.Документ;
      Список.ДобавитьЗначение(Док, "" + Док + " — Остаток = " + Рег.Долг);
      Если Ном = 0 Тогда
         ВыбДок = Док;
      КонецЕсли;
```

```
      Ном = Ном + 1;
   КонецЦикла;
   Если Ном > 0 Тогда
      Если Список.ВыбратьЗначение(ВыбДок, "Выберите кредит") = 1 Тогда
         ДокКредита = ВыбДок;
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: ВыбратьИтоги

ВыгрузитьИтоги

Выгрузить все итоги регистра с текущим фильтром в таблицу значений.

Синтаксис:

```
ВыгрузитьИтоги(<ТаблЗнач>, <ВключатьИзмерФильтра>, <ПредварительноОчищать>)
```

Англоязычный синоним:

```
RetrieveTotals
```

Параметры:

<ТаблЗнач>	Объект типа ТаблицаЗначений, куда система выгрузит все итоги регистра.
<ВключатьИзмерФильтра>	Необязательный параметр. Число: 1 — в получаемую таблицу включаются измерения, закрепленные фильтром; 0 — не включаются. Значение по умолчанию — 0.
<ПредварительноОчищать>	Необязательный параметр. Число: 1 — перед выгрузкой таблица значений очищается; 0 — не очищается. Значение по умолчанию — 1.

Описание:

Метод ВыгрузитьИтоги выгружает все итоги регистра с текущим фильтром (см. УстановитьФильтр) в таблицу значений.

Пример:

```
Функция ВыбратьКредит(ВыбКлиент)
   Список = СоздатьОбъект("ТаблицаЗначений");
   Рег = СоздатьОбъект("Регистр.ТоварныйКредит");
   Рег.УстановитьФильтр(ВыбКлиент, );
   Рег.ВыгрузитьИтоги(Список, 1);
   Возврат Список;
КонецФункции
```

См. также: УстановитьФильтр

ВременныйРасчет

Установить флаг участия регистра во временном расчете.

Синтаксис:

```
ВременныйРасчет(<Флаг>)
```

Англоязычный термин:

```
TempCalc
```

Параметры:

<Флаг>	Необязательный параметр. Числовое выражение. 1 — установить флаг участия регистра во временном расчете; 0 — сбросить флаг участия регистра во временном расчете. Значение по умолчанию — 1.

Возвращаемое значение:

Текущее числовое значение флага участия регистра во временном расчете.

Описание:

Метод ВременныйРасчет устанавливает флаг участия регистра во временном расчете. После такой установки можно выполнять процедуры рассчитатьРегистрыНа и РассчитатьРегистрыПо, тогда у данного регистра методы обращения к остаткам будут выдавать рассчитанные значения остатков на заданный момент времени.

Замечание. В один момент времени только по одному объекту регистров каждого вида могут участвовать во временном расчете.

Пример:

```
// если итоги не актуальны, то стоимости берем из временного расчета
// регистра УчетРеализации
Рег = СоздатьОбъект("Регистр.УчетРеализации");
Рег.ВременныйРасчет();
```

```
РассчитатьРегистрыНа(ТекущийДокумент());
Рег.Товар = Товар;
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

См. также: РассчитатьРегистрыНа, РассчитатьРегистрыПо

УстановитьФильтр

Назначить фильтр для оптимизации действий с регистром.

Синтаксис:

```
УстановитьФильтр(<Измер1>, <Измер2>..., <Рекв1>, <Рекв2>, ...)
```

Англоязычный синоним:

```
SetFilter
```

Параметры:

<Измер1 >, <Измер2>	Выражения со значениями измерений регистра.
<Рекв1>, <рекв2>	Необязательные параметры. Выражения со значениями реквизитов регистра. Заданные значения реквизитов будут влиять только на отбор движений регистра.

Описание:

Метод УстановитьФильтр назначает фильтр для оптимизации действий с регистром (временные расчеты, выбор движений и итогов).

В качестве фильтра задаются конкретные значения измерений и реквизитов регистра. Могут быть заданы не все значения, а только несколько, но обязательно в правильной последовательности, соответствующей структуре выбранного регистра, как она задана в конфигураторе. Измерения и реквизиты могут задаваться с пропусками (неуказанное значение — просто запятая), фиксируются только указанные значения.

Пример:

* Для регистра с измерениями «Товар», «Склад», «МОЛ» запись

```
Рег.УстановитьФильтр(Тов1, , );
```

означает, что расчеты будут проводиться только по одному товару (Тов1), но по всем складам и МОЛам.

```
*
Рег = СоздатьОбъект("Регистр.Взаиморасчеты");
Рег.ВременныйРасчет();
Рег.УстановитьФильтр(Клиент, );
РассчитатьРегистрыНа(ДатаНачала);
Рег.ВыбратьДвиженияСОстатками(ДатаКонца);
Пока Рег.ПолучитьДвижение() = 1 Цикл
   Прих = 0;
   Расх = 0;
   Если Рег.Приход = 1 Тогда
      Прих = Рег.Долг;
   Иначе
      Расх = Рег.Долг;
   КонецЕсли;
   ТекущееСальдо = Рег.Остаток(Клиент, "Задолженность");
   Док = Рег.ТекущийДокумент();
   Таб.ВывестиСекцию("Документ");
КонецЦикла;
```

См. также: ВременныйРасчет

УстановитьЗначениеФильтра

Установить значение фильтра для оптимизации расчетов.

Синтаксис:

```
УстановитьЗначениеФильтра(<ИзмерИлиРеквизит>, <Значен>, <Вариант>)
```

Англоязычный синоним:

```
SetFilterValue
```

Параметры:

<ИзмерИлиРеквизит>	Идентификатор измерения или реквизита.
<Значен>	Значение или список значений.
<Вариант>	Необязательный параметр. Число: 0 — не фильтровать; 1 — фильтровать по значению; 2 — искать вхождение. Значение по умолчанию 1.

Для варианта «2»:

- если параметр <Значен> — это группа справочника, то осуществляется проверка вхождения в группу;
- если параметр <Значен> — это простой элемент справочника или другой тип значения, то осуществляется просто фильтрация по значению;
- для списка значений осуществляется проверка вхождения в список;
- если пустое значение или пустой список значений, то условие не проверяется.

Описание:

Метод УстановитьЗначениеФильтра назначает фильтр для оптимизации расчетов. В качестве фильтра для измерения или реквизита регистра может задаваться либо конкретное значение, либо список значений либо группа для справочника.

Пример:

* Для регистра с измерениями «Товар», «Склад», «МОЛ» запись

```
Рег.УстановитьЗначениеФильтра("Товар", ВыбГруппаТоваров, 2);
```

означает, что расчеты будут проводиться по товарам из группы ВыбГруппаТоваров.

Методы контекста Модуля документа

Следующие методы доступны только в контексте Модуля документа в системной предопределенной процедуре ОбработкаПроведения.

Замечание. Перечисленные в этом разделе методы можно применять только к регистрам, непосредственно доступным в контексте Модуля документа, т. е. нельзя создать объект типа «регистр» с помощью функции СоздатьОбъект и к нему применять метод из данного раздела.

ПривязыватьСтроку

Записать номер строки документа в движении регистра.

Синтаксис:

```
ПривязыватьСтроку(<НомерСтроки>)
```

Англоязычный синоним:

```
LinkLine
```

Параметры:

<НомерСтроки> Выражение, содержащее номер строки документа

Описание:

Метод ПривязыватьСтроку устанавливает, что при выполнении всех последующих движений регистров, движения будут записываться с номером строки <НомерСтроки>.

Метод ПривязыватьСтроку может быть использован как системная процедура: тогда он будет устанавливать номер строки для всех регистров по которым выполняются движения, или как метод конкретного регистра: тогда он будет устанавливать номер строки только для данного регистра.

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока(ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

ДвижениеПриход

Выполнить запись прихода в регистр по параметрам.

Синтаксис:

```
ДвижениеПриход (<Измерен1>, <Измерен2> ... , <Ресурс1>, <Ресурс2> ...)
```

Англоязычный синоним:

ActIncome

Параметры:

<Измерен1>, <Измерен2>	Выражения со значениями измерений регистра.
<Ресурс1>, <Ресурс2>	Выражения со значениями ресурсов регистра.

Описание:

Метод ДвижениеПриход выполняет запись прихода в регистр для текущего документа. Измерения регистра, по которым записывается приход передаются параметрами <Измерение1>, <Измерение2>, ... Величина приращений приходуемых ресурсов регистра передаются параметрами <Ресурс1>, <Ресурс2>...

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Данный метод используется только для регистров остатков.

Пример:

```
Процедура ОбработкаПроведения();
   ВыбратьСтроки();
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ТовЗап.ПривязыватьСтроку(НомерСтроки);
      Регистр.ТовЗап.ДвижениеПриход(Склад, Товар, Количество, Стоимость);
   КонецЦикла;
КонецПроцедуры;
```

См. также: ОбработкаПроведения

ДвижениеРасход

Выполнить запись расхода в регистр по параметрам.

Синтаксис:

ДвижениеРасход(<Измерен1>, <Измерен2>..., <Ресурс1>, <Ресурс2>...)

Англоязычный синоним:

ActOutcome

Параметры:

<Измерен1>, <Измерен2>	Выражения со значениями измерений регистра.
<Ресурс1 >, <Ресурс2>	Выражения со значениями ресурсов регистра.

Описание:

Метод ДвижениеРасход выполняет запись расхода в регистр для текущего документа. Измерения регистра, по которым записывается расход передаются <Измервн1>, <Измерен2>, .. Величина приращений расходуемых ресурсов регистра передаются параметрами <Ресурс1>, <Ресурс2>...

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Данный метод используется только для регистров остатков.

Пример:

```
Процедура ОбработкаПроведения();
   ВыбратьСтроки();
   Пока( ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.ДвижениеРасход(Склад, Товар, Количество, Стоимость);
   КонецЦикла;
КонецПроцедуры
```

См. также: ОбработкаПроведения

ДвижениеПриходВыполнить

Выполнить запись прихода в регистр по атрибутам.

Синтаксис:

ДвижениеПриходВыполнить()

Англоязычный синоним:

DoActIncome

Описание:

Метод ДвижениеПриходВыполнить выполняет запись прихода в регистр для текущего документа. Значения измерений регистра, по которым записывается приход, а также величины приращений приходуемых ресурсов регистра определяются текущими значениями атрибутов регистра.

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Данный метод используется только для регистров остатков.

Пример:

```
Процедура ОбработкаПроведения();
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока(ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеПриходВыполнить();
   КонецЦикла;
КонецПроцедуры;
```

См. также: ОбработкаПроведения

ДвижениеРасходВыполнить

Выполнить запись расхода в регистр по атрибутам.

Синтаксис:

```
ДвижениеРасходВыполнить()
```

Англоязычный синоним:

```
DoActOutcome
```

Описание:

Метод ДвижениеРасходВыполнить выполняет запись расхода в регистр для текущего документа. Значения измерений регистра, по которым записывается приход, а также величины приращений приходуемых ресурсов регистра определяются текущими значениями атрибутов регистра.

Данный метод доступен только в контексте Модуле документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Данный метод используется только для регистров остатков.

Пример:

```
Процедура ОбработкаПроведения()
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока(ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Количество = Количество;
      Регистр.ТовЗап.Стоимость = Стоимость;
      Регистр.ТовЗап.ДвижениеРасходВыполнить();
   КонецЦикла;
КонецПроцедуры
```

См. также: ОбработкаПроведения

Движение

Выполнить запись движения в оборотный регистр по параметрам.

Синтаксис:

```
Движение(<Измерен1>, <Измерен2>..., <Ресурс1> , <Ресурс2>...)
```

Англоязычный синоним:

```
Act
```

Параметры:

<Измерен1 >, <Измерен2>	Выражения со значениями измерений регистра.
<Ресурс1>, <Ресурс2>	Выражения со значениями ресурсов регистра.

Описание:

Метод Движение выполняет запись движения в оборотный регистр для текущего документа. Измерения регистра, по которым записывается движение передаются в параметрах <Измерен1>, <Измерен2>, .. Величины приращений ресурсов регистра передаются параметрами <Ресурс1>, <Ресурс2>...

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. «Работа с Документами»).

Данный метод используется только для оборотных регистров.

Пример:

```
Процедура ОбработкаПроведения();
   ВыбратьСтроки();
   Пока ПолучитьСтроку() > 0 Цикл
      Регистр.ОборотыТоваров.Движение(Товар, Склад, ОборотТовара);
   КонецЦикла;
КонецПроцедуры
```

См. также: ОбработкаПроведения

ДвижениеВыполнить

Выполнить запись движения в оборотный регистр по атрибутам.

Синтаксис:

```
ДвижениеВыполнить()
```

Англоязычный синоним:

```
DoAct
```

Описание:

Метод ДвижениеВыполнить выполняет запись прихода в регистр для текущего документа. Значения измерений регистра, по которым записывается движение, а также величины приращений ресурсов регистра определяются текущими значениями атрибутов регистра.

Данный метод доступен только в контексте Модуля документа в предопределенной процедуре ОбработкаПроведения (см. Гл. "Работа с Документами").

Данный метод используется только для оборотных регистров.

Пример:

```
Процедура ОбработкаПроведения();
   Регистр.ТовЗап.Склад = Склад;
   ВыбратьСтроки();
   Пока (ПолучитьСтроку() > 0) Цикл
      Регистр.ТовЗап.Товар = Товар;
      Регистр.ТовЗап.Оборот = Оборот;
      Регистр.ТовЗап.ДвижениеВыполнить();
   КонецЦикла;
КонецПроцедуры;
```

См. также: ОбработкаПроведения

Системные процедуры работы с регистрами

Нижеследующие процедуры РассчитатьРегистрыНа и РассчитатьРегистрыПо являются глобальными, т. к. они доступны в любом программном модуле и воздействуют на все регистры, объявленные в конфигураторе, у которых установлен флаг участия во временном расчете (см. ВременныйРасчет). Применение данных процедур в тексте программного модуля непосредственное, т. е. просто имя процедуры с параметрами (без предварительной ссылки на объект).

РассчитатьРегистрыНа

Рассчитать все временные регистры на начало события.

Синтаксис:

```
РассчитатьРегистрыНа(<ГраницаРасчета>, <ГрафаОтбора>)
```

Англоязычный синоним:

```
CalcRegsOnBeg
```

Параметры:

<ГраницаРасчета>	Выражение со значением типа дата, документ или позиция документа.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора. "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:

Процедура РассчитатьРегистрыНа рассчитывает все регистры, у которых установлен флаг участия во временном расчете (см. ВременныйРасчет), на момент начала события. Если при вызове метода в параметре передана дата, то расчет временных регистров производится на начало указанной даты. Если при вызове метода в параметре передан документ, то расчет временных регистров производится на момент до проведения данного документа.

Применение данной процедур в тексте любого программного модуля непосредственное, т. е. просто имя процедуры с параметрами (без предварительной ссылки на объект).

Пример:

```
// если итоги не актуальны, то стоимости берем из временного расчета
// регистра УчетРеализации
Рег = СоздатьОбъект("Регистр.УчетРеализации");
Рег.ВременныйРасчет();
РассчитатьРегистрыНа(ТекущийДокумент());
Рег.Товар = Товар;
```

```
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

РассчитатьРегистрыПо

Рассчитать все регистры на конец события.

Синтаксис:

```
РассчитатьРегистрыПо(<ГраницаРасчета>, <ГрафаОтбора>)
```

Англоязычный синоним:

```
CalcRegsOnEnd
```

Параметры:

<ГраницаРасчета>	Выражение со значением типа дата, документ или позиция документа.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора. "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:

Процедура `РассчитатьРегистрыПо` рассчитывает все регистры, у которых установлен флаг участия во временном расчете (см. `ВременныйРасчет`), на момент окончания события. Если при вызове метода в параметре передана дата, то расчет временных регистров производится на конец указанной даты. Если при вызове метода в параметре передан документ, то расчет временных регистров производится на момент после проведения данного документа.

Применение данной процедуры в тексте любого программного модуля непосредственное, т. е. просто имя процедуры с параметрами (без предварительной ссылки на объект).

Пример:

```
// если итоги не актуальны, то стоимости берем из временного расчета
// регистра УчетРеализации
Рег = СоздатьОбъект("Регистр.УчетРеализации");
Рег.ВременныйРасчет();
РассчитатьРегистрыПо(ТекущийДокумент());
Рег.Товар = Товар;
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

Вспомогательный объект Регистры

В системе 1С:Предприятие есть возможность создавать вспомогательные объекты типа «Регистры». Использование этих объектов дает возможность преодолеть ограничение на выполнение временного расчета в один момент времени только по одному регистру каждого вида. Система позволяет выполнять параллельно временные расчеты одних и тех же регистров, но для разных объектов «Регистры». Дело в том, что регистры одного объекта воспринимаются системой независимо от регистров другого объекта.

Во всех программных модулях доступ к атрибутам и методам объекта типа «Регистры» может выполняться при помощи переменной со ссылкой на объект, который создается функцией `СоздатьОбъект`. Чтобы вызвать атрибут или метод объекта, имя этого атрибута или метода (с указанием необходимых параметров) пишется через точку после имени объекта. При создании объекта типа «Регистры» используется ключевое слово `Регистры`. Англоязычный синоним ключевого слова `Регистры` — `Registers`.

Пример:

```
СпецРег = СоздатьОбъект("Регистры");
```

Объект «Регистры» через свои атрибуты предоставляет доступ к любому виду регистра конфигурации. Методы данного объекта позволяют выполнять временный расчет именно для регистров данного объекта, независимо от реги стров другого (аналогичного) объекта.

Атрибуты объекта Регистры

<ИдентификаторРегистра>

Регистр конфигурации.

Синтаксис:
<ИдентификаторРегистра>
Описание:
Атрибут <ИдентификаторРегистра> предоставляет доступ к регистру конфигурации конкретного вида. В тексте программного модуля используется название конкретного регистра, как он назван в конфигураторе.
Пример:
```
СпецРег = СоздатьОбъект("Регистры");
СпецРег.УчетТоваров.СводныйОстаток(ВыбТовар, ВыбСклад, "Количество")
```

Методы объекта Регистры

ПолучитьАтрибут

Получить значение атрибута по идентификатору.
Синтаксис:
ПолучитьАтрибут(<ИмяАтрибута>)
Англоязычный синоним:
GetAttrib
Параметры:

<ИмяАтрибута>	Строковое выражение, содержащее идентификатор регистра конфигурации.

Возвращаемое значение:
Значение атрибута <ИмяАтрибута>.
Описание:
Метод ПолучитьАтрибут позволяет получить значение регистра по идентификатору, как он задан в метаданных.
Пример:
```
СпецРег = СоздатьОбъект("Регистры");
Рег = СпецРег.ПолучитьАтрибут("ОстаткиТоваров");
```

РассчитатьРегистрыНа

Рассчитать все временные регистры на начало события.
Синтаксис:
РассчитатьРегистрыНа(<ГраницаРасчета>, <ГрафаОтбора>)
Англоязычный синоним:
CalcRegsOnBeg
Параметры:

<ГраницаРасчета>	Выражение со значением типа дата, документ или позиция документа.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора. "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:
Процедура РассчитатьРегистрыНа рассчитывает для объекта типа «Регистры» все регистры, у которых установлен флаг участия во временном расчете (см. ВременныйРасчет), на момент начала события. Если при вызове метода в параметре передана дата, то расчет временных регистров производится на начало указанной даты. Если при вызове метода в параметре передан документ, то расчет временных регистров производится на момент до проведения данного документа.
Пример:
```
// если итоги не актуальны, то стоимости берем из временного расчета
// регистра УчетРеализации
СпецРег = СоздатьОбъект("Регистры");
Рег = СпецРег.УчетРеализации;
Рег.ВременныйРасчет();
СпецРег.РассчитатьРегистрыНа(ТекущийДокумент());
Рег.Товар = Товар;
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

РассчитатьРегистрыПо

Рассчитать все регистры на конец события.

Синтаксис:

```
РассчитатьРегистрыПо(<ГраницаРасчета>, <ГрафаОтбора>)
```

Англоязычный синоним:

```
CalcRegsOnEnd
```

Параметры:

<ГраницаРасчета>	Выражение со значением типа дата, документ или позиция документа.
<ГрафаОтбора>	Необязательный параметр. Строковое выражение — идентификатор графы отбора, как он задан в конфигураторе. Данный параметр устанавливает режим использования определенной графы отбора "*" — автоматический выбор графы отбора. Пустая строка — не использовать графу отбора. По умолчанию устанавливается автоматический выбор графы отбора.

Описание:

Процедура `РассчитатьРегистрыПо` рассчитывает для объекта типа «Регистры» все регистры, у которых установлен флаг участия во временном расчете (см. `ВременныйРасчет`), на момент окончания события. Если при вызове метода в параметре передана дата, то расчет временных регистров производится на конец указанной даты. Если при вызове метода в параметре передан документ, то расчет временных регистров производится на момент после проведения данного документа.

Пример:

```
// если итоги не актуальны, то стоимости берем из временного расчета
// регистра УчетРеализации
СпецРег = СоздатьОбъект("Регистры");
Рег = СпецРег.УчетРеализации;
Рег.ВременныйРасчет();
СпецРег.РассчитатьРегистрыПо(ТекущийДокумент());
Рег.Товар = Товар;
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

Актуальность

Устанавливает флаг актуальности временного расчета.

Синтаксис:

```
Актуальность(<ФлагАктуальности>)
```

Англоязычный синоним:

```
Actual
```

Параметры:

<ФлагАктуальности>	Необязательный параметр. Число: 1 — временный расчет поддерживать в актуальном состоянии; 0 — не поддерживать актуальность временного расчета. Если параметр не задан, то метод просто возвращает текущий флат актуальности не меняя его.

Возвращаемое значение:

Текущее состояние флага актуальности временного расчета. Число: 1 — временный расчет поддерживается в актуальном состоянии; 0 — не поддерживается.

Описание:

Метод `Актуальность` устанавливает для объекта типа «Регистры» флаг актуальности временного расчета. Если флаг установлен, то все последующие движения регистров будут изменять итоги временного расчета, а значит итоги регистров временного расчета будут все время (при проведении документа) находиться в актуальном состоянии.

Внимание. Данный метод можно использовать только в модуле проведения документа.

Пример:

```
// стоимости берем из временного расчета
// регистра УчетРеализации
СпецРег = СоздатьОбъект("Регистры");
СпецРег.Актуальность(1);
Рег = СпецРег.УчетРеализации;
Рег.ВременныйРасчет();
```

```
СпецРег.РассчитатьРегистрыНа(ТекущийДокумент());
Рег.Товар = Товар;
Рег.Клиент = Клиент;
Рег.ОстаткиПолучить();
ТекОстаток = Рег.ОстатокТовара;
ТекСтоимость = Рег.Стоимость;
ТекПродСтоимость = Рег.ПродСтоимость;
```

Глава 15
Служебные типы данных компоненты «Бухгалтерский учет»

Тип данных «ПланСчетов»

Тип данных «ПланСчетов» является служебным типом данных. Он предназначен для идентификации Плана счетов, созданного в метаданных. В основном он используется для передачи в качестве параметра различным процедурам и функциям компоненты «Бухгалтерский учет» и для выбора плана счетов в формах. Тип значения «ПланСчетов» не поддерживает никаких данных в информационной базе, а список возможных значений этого типа данных определен планами счетов созданными в конфигурации.

Значения типа «ПланСчетов» могут выступать как реквизиты диалога формы, как реквизиты документов, справочников и т. д.

Для получения значения такого типа данных используется глобальный атрибут `ПланыСчетов`, который имеет в свою очередь набор атрибутов типа «ПланСчетов» соответсвующих имеющимся в конфигурации планам счетов. Кроме того, глобальный атрибут `ПланыСчетов` имеет методы для обхода всех существующих планов счетов.

Методы типа данных «ПланСчетов»

Выбран

Проверяет наличие значения.
Синтаксис:
`Выбран()`
Англоязычный синоним:
`Selected`
Возвращаемое значение:
Числовое значение: 1 — значение не пусто; 0 — значение пусто.
Описание:
Данный метод позволяет определить — пусто значение типа «ПланСчетгт или нет.
Пример:
* `ИспПланСчетов` — является реквизитом диалога формы, имеет тип «План Счетов».

```
Если ИспПланСчетов.Выбран() = 1 Тогда
   БухИтоги.ИспользоватьПланСчетов(ИспПланСчетов);
КонецЕсли;
```

ПорядковыйНомер

Получить порядковый номер плана счетов.
Синтаксис:
`ПорядковыйНомер()`
Англоязычный синоним:
`Number`
Возвращаемое значение:
Числовое значение: порядковый номер плана счетов среди планов счетов конфигурации.
Описание:
Метод `ПорядковыйНомер` позволяет определить порядковый номер плана счетов среди планов счетов, созданных в конфигурации.
Пример:

```
Если ИспПланСчетов.Выбран() = 1 Тогда
   Сообщить("План счетов" + ИспПланСчетов.ПорядковыйНоиер());
КонецЕсли;
```

Идентификатор

Определить строку-идентификатор плана счетов как он задан в метаданных.
Синтаксис:
`Идентификатор()`
Англоязычный синоним:
`Identifier`
Возвращаемое значение:
Строка-идентификатор плана счетов как он задан в метаданных.

Описание:

Метод Идентификатор позволяет определить строку-идентификатор плана счетов как он задан в метаданных.

Пример:

```
Если ИспПланСчетов.Выбран() = 1 Тогда
   Сообщить(Идентификатор(ИспПланСчетов));
КонецЕсли;
```

Атрибут глобального контекста «ПланыСчетов»

ПланыСчетов

Атрибут глобального контекста для получения существующих значений типа «ПланСчетов».

Синтаксис:

ПланыСчетов

Англоязычный синоним:

Charts0fAccounts

Описание:

Атрибут ПланыСчетов не имеет самостоятельного смысла, а служит для получения конкретных значений типа «ПланСчетов». Атрибут всегда используется с доступными ему атрибутами и методами.

Атрибуты и методы:

Для получения конкретного значения типа «План счетов» следует через точку указать атрибут — идентификатор плана счетов.

Для получения количества существующих в конфигурации планов счетов следует через точку вызвать метод КоличествоЗначений.

Для получения значения типа «План счетов» по его номеру в метаданных, следует через точку вызвать метод ЗначениеПоНомеру(<Число>), где <Число> — номер плана счетов в метаданных.

Пример:

```
Сч = СоздатьОбъект("Счет");
// ...
Если Сч.ПланСчетов() = ПланыСчетов.РабПлан Тогда
   // ...
КонецЕсли;
Для Инд = 1 По ПланыСчетов.КоличествоЗначений() Цикл
   ПлСч = ПланыСчетов.ЗначениеПоНомеру(Инд);
КонецЦикла;
```

Методы глобального атрибута «ПланыСчетов»

Значения типа «ПланСчетов» могут выступать как реквизиты диалога формы, как реквизиты документов, справочников и т. д.

Для получения значений типа «ПланСчетов» используется глобальный атрибут ПланыСчетов, который имеет в свою очередь набор атрибутов типа «План Счетов» соответсвующих имеющимся в конфигурации планам счет Кроме того, глобальный атрибут ПланыСчетов имеет методы для обхода в существующих планов счетов.

КоличествоЗначений

Получить общее количество планов счетов.

Синтаксис:

КоличествоЗначений()

Англоязычный синоним:

Count

Возвращаемое значение:

Числовое значение: количество планов счетов конфигурации.

Описание:

Метод КоличествоЗначений применяется к атрибуту глобального контекста ПланыСчетов и позволяет определить общее количество планов счетов созданных в конфигурации.

Пример:

```
Для Инд=1 По ПланыСчетов.КоличествоЗначений() Цикл
   ПлСч = ПланыСчетов.ЗначениеПоНомеру(Инд);
КонецЦикла;
```

ЗначениеПоНомеру

Определить план счетов, по номеру в списке планов счетов конфигурации.
Синтаксис:
`ЗначениеПоНомеру(<Номер>)`
Англоязычный синоним:
`ValueByIndex`
Параметры:
`<Номер>` Числовое выражение. Номер позиции плана счетов, заданный в Конфигураторе.
Возвращаемое значение:
Значение типа «План Счетов», соответствующее номеру заданной позиции.
Описание:
Метод `ЗначениеПоНомеру` применяется к атрибуту глобального контекста `ПланыСчетов` и позволяет определить план счетов, соответствующий номеру в списке планов счетов конфигурации.
Пример:
```
Для Инд = 1 По ПланыСчетов.КоличествоЗначений() Цикл
   ПлСч = ПланыСчетов.ЗначениеПоНомеру(Инд);
КонецЦикла
```

ЗначениеПоИдентификатору

Определить план счетов, по идентификатору.
Синтаксис:
`ЗначениеПоИдентификатору(<Идентификатор>)`
Англоязычный синоним:
`ValueByIdentifier`
Параметры:
`<Идентификатор>` Строковое выражение. Идентификатор плана счетов, заданный в конфигураторе.
Возвращаемое значение:
Значение плана счетов, соответствующее идентификатору в метаданных. Если не найдено — то пустое значение.
Описание:
Метод `ЗначениеПоИдентификатору` применяется к атрибуту глобального контекста `ПланыСчетов` и позволяет определить план счетов, соответствующий идентификатору плана счетов конфигурации.
Пример:
```
ПлСч = ПланыСчетов.ЗначениеПоИдентификатору("Основной");
```

Тип данных «ВидСубконто»

Тип данных «ВидСубконто» является служебным типом данных. Он предназначен для идентификации самого Вида субконто. созданного в метаданных. В основном он используется для передачи в качестве параметра различным процедурам и функциям компоненты «Бухгалтерский учет» и для выбора вида субконто в формах. Тип данных «ВидСубконто» не поддерживает никаких данных в информационной базе, а список возможных значений этого типа данных определен видами субконто, созданными в конфигурации.

Значения типа «ВидСубконто» могут выступать как реквизиты диалога формы, как реквизиты документов, справочников и т. д.

Для получения значения такого типа используется глобальный атрибут `ВидыСубконто`, который имеет в свою очередь набор атрибутов типа «ВидСубконто» соответсвующих имеющимся видам субконто. Кроме того, глобальный атрибут `ВидыСубконто` имеет методы для обхода всех существующих видов субконто.

Методы типа данных «ВидСубконто»

Выбран

Проверяет наличие значения.
Синтаксис:
`Выбран()`
Англоязычный синоним:
`Selected`
Возвращаемое значение:
Числовое значение:
1 — значение не пусто;
0 — значение пусто.
Описание:

Данный метод позволяет определить — пусто значение типа «ВидСубконто» или нет.

Пример:

* ВыбВидСубк1 — является реквизитом диалога формы типа «ВидСубконто».

```
Если ВыбВидСубк1.Выбран() = 1 Тогда
   БухИтоги.ИспользоватьСубконто(ВыбВидСубк1);
КонецЕсли;
```

ПорядковыйНомер

Получить порядковый номер вида субконто.

Синтаксис:

```
ПорядковыйНомер()
```

Англоязычный синоним:

```
Number
```

Возвращаемое значение:

Числовое значение: порядковый номер вида субконто среди видов.

Описание:

Метод ПорядковыйНомер позволяет определить порядковый номер вида субконто среди видов субконто, созданных в конфигурации.

Пример:

```
Если ВыбВидСубк1.Выбран() = 1 Тогда
   Сообщить("Вид номер" + ВыбВидСубк1.ПорядковыйНомер());
КонецЕсли;
```

ТипСубконто

Получить тип субконто.

Синтаксис:

```
ТипСубконто()
```

Англоязычный синоним:

```
SubcontoType
```

Возвращаемое значение:

Строковое значение, описывающее тип субконто, например "Справочник.Контрагенты".

Описание:

Метод ТипСубконто позволяет определить тип субконто, как он определен в конфигурации.

Пример:

```
Сообщить("Тип субконто " + ВыбВидСубк1.ТипСубконто());
```

Идентификатор

Определить строку-идентификатор вида субконто как он задан в метаданных.

Синтаксис:

```
Идентификатор()
```

Англоязычный синоним:

```
Identifier
```

Возвращаемое значение:

Строка-идентификатор вида субконто как он задан в метаданных.

Описание:

Метод Идентификатор позволяет определить строку-идентификатор вида субконто как он задан в метаданных.

Пример:

```
Для Н=1 По ВидыСубконто.КоличествоЗначений() Цикл
   Сообщить(Идентификатор(ВидыСубконто.ЗначениеПоНомеру(Н)));
КонецЦикла;
```

Атрибут глобального контекста «ВидыСубконто»

ВидыСубконто

Атрибут глобального контекста для получения существующих значений типа ВидСубконто.

Синтаксис:

```
ВидыСубконто
```

Англоязычный синоним:

```
SubcontoKinds
```

Описание:

Атрибут ВидыСубконто не имеет самостоятельного смысла, а служит для получения конкретных значений типа «Вид субконто». Атрибут всегда используется с доступными ему атрибутами и методами.

Атрибуты и методы:

Для получения конкретного значения типа «Вид субконто» следует через точку указать атрибут-идентификатор вида субконто.

Для получения количества существующих в конфигурации видов субконто следует через точку вызвать метод КоличествоЗначений.

Для получения значения типа «Вид субконто» по его номеру в метаданных следует через точку вызвать метод ЗначениеПоНомеру(<Число>), где <Число> — номер вида субконто в метаданных.

Пример:

```
Сч = СоздатьОбъект("Счет");
Если Сч.ВидСубконто(1) = ВидыСубконто.Материалы Тогда
   // ...
КонецЕсли;
Для Инд = 1 По ВидыСубконто.КоличествоЗначений() Цикл
   ВидСк = ВидыСубконто.ЗначениеПоНомеру(Инд);
   // ...
КонецЦикла;
```

Методы глобального атрибута «ВидыСубконто»

Для получения значений типа «ВидСубконто» используется глобальный атрибут ВидыСубконто, который имеет в свою очередь набор атрибутов типа «ВидСубконто» соответсвующих имеющимся видам субконто. Кроме того, глобальный атрибут ВидыСубконто имеет методы для обхода всех существующих видов субконто.

КоличествоЗначений

Получить общее количество видов субконто.

Синтаксис:

КоличествоЗначений()

Англоязычный синоним:

Count

Возвращаемое значение:

Числовое значение: количество видов субконто конфигурации.

Описание:

Метод КоличествоЗначений применяется к атрибуту глобального контекста ВидыСубконто и позволяет определить общее количество видов субконто, созданных в конфигурации.

Пример:

```
Для Инд = 1 По ВидыСубконто.КоличествоЗначений() Цикл
   ВидСк = ВидыСубконто.ЗначениеПоНомеру(Инд);
КонецЦикла;
```

ЗначениеПоНомеру

Определить вид субконто, по номеру в списке видов субконто конфигурации.

Синтаксис:

ЗначениеПоНомеру(<Номер>)

Англоязычный синоним:

ValueByIndex

Параметры:

<Номер> Числовое выражение. Номер позиции вида субконто, заданный в конфигураторе.

Возвращаемое значение:

Значение типа «ВидСубконто», соответствующее номеру заданной позиции.

Описание:

Метод ЗначениеПоНомеру применяется к атрибуту глобального контекста ВидыСубконто и позволяет определить вид субконто, соответствующий номеру в списке видов субконто конфигурации.

Пример:

```
Для Инд = 1 По ВидыСубконто.КоличествоЗначений() Цикл
   ВидСк = ВидыСубконто.ЗначениеПоНомеру(Инд);
КонецЦикла;
```

ЗначениеПоИдентификатору

Определить вид субконто, по идентификатору видов субконто конфигурации.

Синтаксис:

```
ЗначениеПоИдентификатору(<Идентификатор>)
```

Англоязычный синоним:

```
ValueByIdentifier
```

Параметры:

<Идентификатор>	Строковое выражение. Идентификатор вида субконто, заданный в конфигураторе.

Возвращаемое значение:

Значение типа «ВидСубконто», соответствующее идентификатору в метаданных. Если не найдено — то пустое значение.

Описание:

Метод `ЗначениеПоИдентификатору` применяется к атрибуту глобального контекста `ВидыСубконто` и позволяет определить вид субконто, соответствующий идентификатору видов субконто конфигурации.

Пример:

```
ВидСк = ВидыСубконто.ЗначениеПоИдентификатору("Контрагент");
```

Глава 16
работа с бухгалтерскими счетами

`Счета` — это агрегатный тип данных для доступа к объектам данных — бухгалтерским счетам. Бухгалтерские счета используются компонентой «Бухгалтерский учет» системы 1С:Предприятие для идентификации разрезов синтетического учета наличия и движения средств. В общем виде смысл типа данных «Счет» вполне соответствует общепринятому понятию «Счет» в бухгалтерском учете. Подробно об основных свойствах типа данных «Счет» можно ознакомиться в руководстве по конфигурированию системы 1С:Предприятие.

В конфигурации системы может быть создано несколько планов счетов. План счетов является фактически видом для значения типа «Счет». Например, реквизит диалога типа «Счет» может иметь конкретный вид (относиться к конкретному плану счетов) или быть неопределенного вида — то есть принимать значение различных планов счетов.

Структура данных для объектов типа «Счет» задается в конфигураторе и является одинаковой для всех планов счетов. Счета имеют стандартные реквизиты (код, наименование и т. д.), для которых в конфигурации настраиваются только их свойства (например, длина наименования). Кроме того, для счетов в конфигурации могут быть заданы дополнительные реквизиты. Дополнительные реквизиты могут быть периодическими, то есть иметь разные значения на разные даты.

Список счетов (план счетов) может быть многоуровневым. При этом каждый объект типа «Счет» вне зависимости от реального наличия подчиненных счетов всегда является либо конкретным счетом, либо группой. Это свойство конкретного счета задается при его создании и не изменяется в дальнейшем. При этом счета-группы не могут участвовать в проводках.

Заметим, что бухгалтерские счета внесенные в конфигураторе в метаданные не могут изменять свои основные реквизиты (код, наименование, настройки аналитического, валютного и количественного учета).

Контекст работы с бухгалтерскими счетами

В синтаксисе языка доступ к атрибутам, а также вызов методов счетов зависит от контекста выполнения программного модуля.

Если счет входит (согласно локальному контексту) в набор непосредственно Доступных модулю значений агрегатных типов данных (см. «Виды программ-йьгх модулей»), то доступ к атрибутам и вызов метода для этого счета — просто имя этого атрибута или метода с указанием необходимых параметров.

Пример:

* Модуль формы счета выполняется в контексте «Счет». Поэтому в модуп возможен непосредственный доступ к текущему счету. Например, для пои сваивания счету наименования запишем:

```
Наименование = "Основные средства";
```

Значение счета может быть получено из других источников, например, как реквизит документа. В этом случае обращение к атрибутам и методам такого документа представляет собой сложное выражение, где имена реквизитов разделяются точкой.

Пример:

* Например, в структуре документа «ПриходныйОрдер» существует реквизит «КоррСчет» типа «Счет». Тогда получить наименование счета, указанного в этом реквизите, можно следующим образом

```
НаименованиеСчета = Док.КоррСчет.Наименование;
```

В других случаях, доступ к атрибутам, вызов методов конкретного счета происходит при помощи переменной со ссылкой на объект типа «Счет». Объект создается функцией `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать атрибут или метод объекта, имя этого атрибута или метода (с указанием необходимых параметров) пишется через точку после имени ссылки.

При создании ссылки на объект типа «Счет» при помощи функции `СоздатьОбъект` в качестве типа объекта указывается слово "Счет" и может быть через точку указан вид — идентификатор плана счетов.

Полное имя типа счет записывается следующим образом:

`Счет.<ВидСчета>`

где **<**`ВидСчета`**>** — идентификатор плана счетов.

Применение ключевого слова "`Счет`" (без вида счета) используется для организации доступа ко всем планам счетов.

Англоязычный синоним ключевого слова `Счет` — `Account`.

Замечание: Следует обратить особое внимание, что переменная типа «Счет», созданная функцией `СоздатьОбъект` — это ссылка на список счетов, в отличие от переменных, содержащих само значение объекта (например, переменной может быть присвоено значение некоторого реквизита документа, который имеет тип «Счет»). Использование ссылки на список счетов, созданной при помощи функции `СоздатьОбъект`, существенно отличается от работы со значением типа «Счет». Только при работе со объектом-ссылкой на список счетов разрешено изменять позицию (найти-выбрать...) текущего счета в списке (т. е. осуществлять позиционирование по списку счетов), создавать новые, изменять и удалять существующие счета, С другой стороны, ссылка на список счетов не содержит собственно

значения конкретного счета, которое можно присвоить чему-либо. Однако. его всегда можно получить, используя функцию ТекущийСчет.

Замечание. Объект, созданный при помощи функции СоздатьОбъект, изначально не определен, т. е. не содержит никакого значения. Чтобы начать с ним работать, его предварительно надо позиционировать (установить на конкретный счет) при помощи процедур НайтиСчет, ПолучитьСчет и т. п.

Пример:

```
*
Сч = СоздатьОбъект("Счет");
Сч1 = СоздатьОбъект("Счет.ОснПлан");
Сч2 = СоздатьОбъект("Счет.РабПлан");
*
//В модуле формы счета // меняем наименование счета
Наименование = "Новый счет";
*
Сч1 = СоздатьОбъект("Счет.ОснПлан");
// создаем новый счет Сч1.Новый();
Сч1.Код = "12.01";
Сч1.Записать();
```

Атрибуты объекта «Счет»

Код

Полный код счета.

Синтаксис:

Код

Англоязычный синоним:

Code

Описание:

Код счета в общем случае представляет собой символьную строку вида:

<Код счета>.<Код субсчета>.<Код субсчета> ...

Общая длина кода счета в системе 1С:Предприятие ограничена 255 символами. В это значение входят: код счета первого уровня, коды субсчетов всех нижележащих уровней и разделители номеров счета и субсчетов (точка). При помощи атрибута Код можно получать и задавать код счета.

Пример:

* При показе готового табличного документа (в данном примере это отчет «Карточка счета») в заголовке окна выдается наименование отчета и код счета, по которому построен отчет.

```
// Формирование отчета "Карточка счета"
Табл = СоздатьОбъект("Таблица");
// ...
<команды формирования отчета и выдачи секций табличного документа>
// ...
// Показ готового отчета
Табл.Показать("Карточка счета: " + Счет.Код);
```

Наименование

Наименование счета.

Синтаксис:

Наименование

Англоязычный синоним:

Description

Описание:

Наименование счета представляет собой произвольную строку символов. Наименование, как правило, разъясняет назначение счета. Максимальная длина наименования счета задается при редактировании свойств планов счетов в Конфигураторе.

При помощи атрибута Наименование можно получать и задавать наименование счета.

Пример:

* При показе готового табличного документа (в данном примере это отчет «Карточка счета») в заголовке окна выдается наименование отчета, код и наименование счета, по которому построен отчет.

```
// Формирование отчета "Карточка счета"
Табл = СоздатьОбъект("Таблица");
// ...
<команды формирования отчета и выдачи секций табличного документа>
// ...
// Показ готового отчета
Табл.Показать("Карточка счета: " + Счет.Код + " " + Счет.Наименование);
```

Валютный

Признак ведения валютного учета.

Синтаксис:

```
Валютный
```

Англоязычный синоним:

```
IsCurrency
```

Значение:

1 — валютный учет ведется по данному счету;
0 — валютный учет не ведется по данному счету;

Описание:

Атрибут содержит признак ведения валютного учета по счету. Изменение данного значения из языка следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч.Валютный = 1;
```

Количественный

Признак ведения количественного учета.

Синтаксис:

```
Количественный
```

Англоязычный синоним:

```
IsAmount
```

Значение:

1 — количественный учет ведется по данному счету;
0 — количественный учет не ведется по данному счету;

Описание:

Атрибут содержит признак ведения количественного учета по счету. Изменение данного значения из языка следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч.Количественный = 1;
```

Забалансовый

Признак забалансовости счета.

Синтаксис:

```
Забалансовый
```

Англоязычный синоним:

```
IsSingle
```

Значение:

1 — выбранный счет является забалансовым счетом;
0 — выбранный счет является балансовым счетом.

Описание:

Атрибут содержит признак того, что счет является забалансовым, то есть не участвует в двойной записи, не требует в проводках наличия корреспонденции и не может корреспондировать с балансовыми счетами. Изменение данного значения из языка следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч.Забалансовый = 1;
```

Активный

Тип остатка по счету (активный, пассивный, активно-пассивный).

Синтаксис:
Активный
Англоязычный синоним:
IsActive
Значение:
1 — счет является активным;
2 — счет является пассивным;
3 — счет является активно-пассивным;
Описание:
Атрибут содержит тип остатка счета. Счет может являться активным, (остатки должны быть дебетовыми), пассивным (остатки должны быть кредитов выми), активно-пассивным (остатки могут быть дебетовыми и кредитовыми).

Данный признак влияет на способ отражения остатков по счету в итогах. Например, для активного счета, превышение кредитового оборота над дебетовым приведет к отрицательному дебетовому остатку, а для активно-пассивного — к положительному кредитовому.

По умолчанию все счета считаются активно-пассивными.
Пример:
```
Сч.Активный = 2;
```

<Реквизит>

Возвращает или задает значение дополнительного реквизита счета.
Синтаксис:
<Реквизит> Идентификатор дополнительного реквизита счета, как он задан в конфигураторе.
Описание:
Помимо реквизитов счета, задаваемых на системном уровне (код, наименование, признаки количественного и валютного учета, и другие), в процессе конфигурирования для счета можно создать практически неограниченное число дополнительных реквизитов для хранения любой необходимой информации.

Атрибут <Реквизит> позволяет обращаться к значениям дополнительных реквизитов выбранного счета. Для обращения к конкретному реквизиту должен быть указан его идентификатор, заданный для этого реквизита в конфигураторе.
Пример:
* Для счетов определен реквизит "Ответственный" типа "Справочник.Сотрудники".
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.ВыбратьСчета();
Пока Сч.ПолучитьСчет() = 1 Цикл
   Сч.Ответственный = ВыбСотр;
   Сообщить(Сч.Ответственный.Наименование);
   Сч.Записать();
КонецЦикла
```

Методы объекта «Счет»

ВыбратьСчета

Метод открывает выборку счетов из текущего плана счетов для последующей обработки.
Синтаксис:
ВыбратьСчета()
Англоязычный синоним:
SelectAccounts
Возвращаемое значение:
Число: 1 — если действие выполнено и в выборке есть хотя бы один счет; 0 — если действие не выполнено или в выборке нет ни одного счета.
Описание:
Метод ВыбратьСчета открывает выборку счетов для текущего плана счетов. Вызов данного метода позволяет затем перебрать счета. Непосредственно извлечение счетов из выборки осуществляется при помощи метода ПолучитьСчет. Счета извлекаются в порядке возрастания кода счета.
Пример:
```
Процедура Сформировать()
   Сч = СоздатьОбъект("Счет.Основной");
   Сч.ВыбратьСчета();
   Пока Сч.ПолучитьСчет() = 1 Цикл
      Сообщить(Сч.Код);
   КонецЦикла;
```

```
КонецПроцедуры
```

ПолучитьСчет

Получить из выборки следующий счет. Выборка должна быть предварительно открыта при помощи метода ВыбратьСчета.

Синтаксис:

```
ПолучитьСчет([<Режим>])
```

Англоязычный синоним:

```
GetAccount
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение — признак выборки подчиненных счетов. Может принимать значения: 0 — не включать подчиненные счета; 1 — включать подчиненные счета. По умолчанию — 1.

Возвращаемое значение:

Число: 1 — следующий счет выбран успешно; 0 — следующий счет не выбран (отсутствует).

Описание:

Метод ПолучитьСчет выбирает очередной счет из выборки, содержащей счета из плана счетов. Перед применением метода ПолучитьСчет выборка должна быть открыта при помощи метода ВыбратьСчета. Счета извлекаются в порядке возрастания кода счета.

Метод ПолучитьСчет используется для организации цикла по счетам. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьСчет возвращает 1.

Метод возвращает 0, когда очередной счет не выбран. Это происходит, если при предыдущем применении метода был выбран последний счет выборки.

Пример:

```
Процедура Сформировать()
   Сч = СоздатьОбъект("Счет.Основной");
   Сч.ВыбратьСчета();
   Пока Сч.ПолучитьСчет() = 1 Цикл
      Сообщить(Сч.Код);
   КонецЦикла;
КонецПроцедуры
```

УстановитьАтрибут

Установить значение реквизита по имени идентификатора.

Синтаксис:

```
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SetAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.
<Значение>	Выражение, содержащее устанавливаемое значение реквизита.

Описание:

Метод УстановитьАтрибут позволяет установить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
Счт.УстановитьАтрибут("ЦенаРозн", ЦенаТов);
```

ПолучитьАтрибут

Получить значение реквизита по имени идентификатора.

Синтаксис:

```
ПолучитьАтрибут(<ИмяРеквизита>)
```

Англоязычный синоним:

```
GetAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.

Возвращаемое значение:

Значение реквизита <ИмяРеквизита>.

Описание:

Метод ПолучитьАтрибут позволяет получить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:
```
ЦенаТов = Счт.ПолучитьАтрибут("ЦенаРозн");
```

Выбрать

Выбрать счет из плана счетов в диалоге.

Синтаксис:
```
Выбрать(<Подсказка>, <ФормаСписка>)
```
Англоязычный синоним:
```
Choose
```
Параметры:

<Подсказка>	Необязательный параметр. Символьное выражение — текст заголовка окна диалога выбора счета. Может использоваться в качестве подсказки.
<ФормаСписка>	Необязательный параметр. Символьное выражение — идентификатор формы списка плана счетов (как он указан в конфигураторе), которая должна использоваться для выбора. Если значение пустое, то используется форма списка для выбора по умолчанию.

Возвращаемое значение:
Число: 1 — счет выбран; 0 — счет не выбран.

Описание:

Метод Выбрать вызывает на экран диалог, содержащий план счетов. Пользователь должен интерактивно выбрать требуемый счет.

Если счет выбран, метод возвращает 1 и позиционирует объект на выбранном счете.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.Выбрать("Выберите счет", "ДляВыбораВДиалоге");
```

НайтиПоКоду

Выполняет поиск счета в плане счетов по коду счета.

Синтаксис:
```
НайтиПоКоду(<КодСчета>)
```
Англоязычный синоним:
```
FindByCode
```
Параметры:

<КодСчета>	Символьное выражение — код счета, как он указан в плане счетов.

Возвращаемое значение:
Число: 1 — счет найден; 0 — счет не найден.

Описание:

Метод НайтиПоКоду выполняет поиск счета в плане счетов по коду счета, который передается методу в качестве параметра.

Если счет найден, метод возвращает 1 и позиционирует объект на выбранном счете.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("01.01");
```

НайтиСчет

Метод выполняет поиск счета в плане счетов по значению типа «Счет».

Синтаксис:
```
НайтиСчет(<Счет>)
```
Англоязычный синоним:
```
FindAccount
```
Параметры:

<Счет>	Значение типа «Счет».

Возвращаемое значение:
Число: 1 — счет найден; 0 — счет не найден.

Описание:

Метод НайтиСчет выполняет поиск счета по значению, заданному параметром <Счет>, и позиционирует объект на выбранном счете.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиСчет(ВыбСчет);
```

Выбран

Проверяет факт выбора счета.
Синтаксис:
```
Выбран()
```
Англоязычный синоним:
```
Selected
```
Возвращаемое значение:
Числ: 1 — если счет выбран (спозиционирован); 0 — если не выбран.
Описание:
Метод `Выбран` для объекта созданного функцией `СоздатьОбъект` определяет, спозиционирован ли объект на каком либо счете или нет, для объекта хранящего значение типа «Счет» он определяет является значение пустым или нет. Метод возвращает число со значением 1 — если счет выбран (спозиционирован) или значение не пустое, 0 — если не выбран или значение пустое.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду ("76.02");
Если Сч.Выбран() = 1 Тогда
   Сообщить("Счет найден");
КонецЕсли;
```

ЗаданВКонфигурации

Определяет, задан текущий счет в конфигурации или в информационной базе.
Синтаксис:
```
ЗаданВКонфигурации()
```
Англоязычный синоним:
```
DefinedInConfiguration
```
Возвращаемое значение:
Число: 1 — если счет задан в конфигурации; 0 — если счет введен в информационной базе.
Описание:
Метод `ЗаданВКонфигурации` определяет, задан текущий счет в конфигурации или в информационной базе.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Если Сч.ЗаданВКонфигурации() = 1 Тогда
   Сообщить("Счет задан в Конфигурации ");
КонецЕсли;
```

Вид

Позволяет определить, к какому плану счетов относится указанный счет.
Синтаксис:
```
Вид()
```
Англоязычный синоним:
```
Kind
```
Возвращаемое значение:
Символьная строка — идентификатор плана счетов, к которому относится данный счет.
Описание:
Метод `Вид` позволяет определить план счетов, к которому относится данный счет. Метод возвращает идентификатор плана счетов в виде строки символов. Метод `ПланСчетов` позволяет получить значение типа «План счетов» к которому относится выбранный счет.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сообщить("Вид - " + Сч.Вид());
```

ПредставлениеВида

Определить пользовательское представление вида плана счетов.
Синтаксис:
```
ПредставлениеВида()
```
Англоязычный синоним:
```
KindPresent
```

Возвращаемое значение:

Строковое значение, содержащее пользовательское представление вида плана счетов (синоним вида плана счетов или, если он пустой, то идентификатор вида).

Описание:

Метод ПредставлениеВида позволяет получить пользовательское представление вида плана счетов, как оно задано в конфигураторе.

Пример:

```
// отобразим пользовательское представление в строке состояния
Сч = СоздатьОбъект("Счет.Основной");
Состояние(Сч.ПредставлениеВида());
```

ПланСчетов

Выдает план счетов, которому принадлежит счет.

Синтаксис:

```
ПланСчетов()
```

Англоязычный синоним:

```
ChartOfAccounts
```

Возвращаемое значение:

Значение типа «План счетов».

Описание:

Метод выдает план счетов (значение типа «План счетов»), которому принадлежит счет.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сообщить("План счетов - " + Сч.ПланСчетов());
// Выдает "План счетов - Основной"
```

ЭтоГруппа

Определяет, является ли данный счет группой.

Синтаксис:

```
ЭтоГруппа()
```

Англоязычный синоним:

```
IsGroup
```

Возвращаемое значение:

Число: 1 — счет является группой; 0 — счет не является группой.

Описание:

Для каждого счета при создании указывается — может ли он иметь субсчета (будет являться группой) или будет собственно счетом (не группой). Счета-группы не могут участвовать в проводках. В дальнейшем это свойство счета не изменяется.

Данная функция определяет, является счет группой (может иметь субсчета) или нет.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Если Сч.ЭтоГруппа() = 1 Тогда
   Сообщить("Найдена группа");
КонецЕсли;
```

Уровень

Возвращает уровень вложенности текущего счета.

Синтаксис:

```
Уровень()
```

Англоязычный синоним:

```
Level
```

Возвращаемое значение:

Число — уровень вложенности счета: 1 — счета верхнего уровня.

Описание:

В системе 1С:Предприятие количество уровней вложенности субсчетов в плане счетов не ограничивается. В качестве «ограничителя» выступает величина реквизита «Максимальная длина кода счета» в окне редактирования «Планы счетов». В пределах указанной длины можно задавать произвольное количество уровней вложенности, комбинируя символы «#» и «.» в шаблоне счета.

Шаблон кода представляет собой символьную строку состоящую из символов «#» и «.». Шаблон кода схематически иллюстрирует структуру полного кода счета. Фактически, шаблон кода задает количество уровней субсчетов, которое может быть открыто к счету. Например, символьная строка вида

```
##.#.###
```

означает, что к счету могут быть открыты 2 уровня субсчетов. Метод `Уровень()` возвращает уровень вложенности текущего счета. Для счета верхнего уровня метод возвратит 1, для счета второго уровня — 2, и так далее.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сообщить("Уровень счета - " + Сч.Уровень());
//Возвращает — 2
```

ТекущийСчет

Возвращает текущее значение счета.

Синтаксис:

```
ТекущийСчет()
```

Англоязычный синоним:

```
CurAccount
```

Возвращаемое значение:

Значение типа «Счет».

Описание:

Объекты созданные функцией `СоздатьОбъект` не содержат собственно значения типа «Счет», так как они могут быть спозиционированы на разные счета. Метод `ТекущийСчет` позволяет получить значение типа «Счет» у такого объекта в соответствии с его текущей позицией. Данный метод применяется, например, если нужно счет передать как параметр в вызове какого-либо метода, или присвоить какому-либо реквизиту, или занести в проводку.

Кроме того, в контексте формы счета данный метод может выдать текущее (для открытой формы) значение типа «Счет».

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Если Сч.Выбран() = 1 Тогда
   Операция.Дебет.Счет = Сч.ТекущийСчет();
КонецЕсли;
```

ПометкаУдаления

Проверяет наличие пометки на удаление для текущего счета.

Синтаксис:

```
ПометкаУдаления()
```

Англоязычный синоним:

```
DeleteMark
```

Возвращаемое значение:

1 — счет помечен на удаление;
0 — счет не помечен на удаление.

Описание;

Метод используется для проверки, помечен ли на удаление текущий счет.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Если Сч.ПометкаУдаления() = 1 Тогда
   Сообщить("Счет помечен на удаление");
КонецЕсли;
```

ВыборГруппы

Устанавливает или сбрасывает признак выбора группы.

Синтаксис:

```
ВыборГруппы([<ПризнакВыбораГруппы>])
```

Англоязычный синоним:

```
SelectGroup
```

Параметры:

`<ПризнакВыбораГруппы>`	Необязательный параметр. Числовое выражение: 1 — выбирать группы; 0 — не выбирать группы.

Возвращаемое значение:
Текущее числовое значение режима выборки групп (на момент до исполнения метода).
Описание:
Метод ВыборГруппы устанавливает режим выборки групп. Данный метод может применяться как для позиционируемых объектов, созданных функцией СоздатьОбъект, так и для элементов диалога типа «Счет» (см. «Методы элементов диалога»).
Рекомендуется отключать возможность выбора группы, если значение будет использовано для формирования проводок.
По умолчанию выбор группы разрешен, за исключением счетов дебета и кредита в проводках операции.
Режим, устанавливаемый данным методом для элементов диалога, влияет на интерактивный выбор пользователем значения типа «Счет» из списка счетов (плана счетов). Такой выбор производится вызовом метода Выбрать или нажатием кнопки выбора, если объект является элементом диалога. Установка режима позволяет разрешить или запретить пользователю выбирать счета-группы (счета, которые могут иметь субсчета). Это имеет смысл для установки режима сразу на все значения реквизита табличной части документа.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.ВыборГруппы(0);
Сч.Выбрать("Выберите конкретный счет");
```

ПринадлежитГруппе

Определяет, является ли текущий счет субсчетом для счета, указанного в качестве параметра метода.
Синтаксис:
ПринадлежитГруппе(<Счет>)
Англоязычный синоним:
BelongsToGroup
Параметры:
<Счет> Значение типа «Счет» — счет, к субсчетам которого проверяется принадлежность текущего счета.
Возвращаемое значение:
Число: 1 — текущий счет является субсчетом для счета, указанного в качестве параметра; 0 — текущий счет не является субсчетом для счета, указанного в качестве параметра.
Описание:
Метод ПринадлежитГруппе позволяет проверить, является ли текущий счет субсчетом для счета, переданного в качестве параметра метода. Проверка выполняется по всем вышестоящим уровням счета.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Если Сч.ПринадлежитГруппе(СчетПоКоду("76")) = 1 Тогда
   Сообщить("Счет принадлежит группе");
КонецЦикла;
```

КоличествоСубконто

Определяет количество субконто у текущего счета.
Синтаксис:
КоличествоСубконто()
Англоязычный синоним:
SubcontoCount
Возвращаемое значение:
Число — количество видов субконто у текущего счета.
Описание:
Максимальное количество видов субконто, которое можно «прикрепить» к счету, устанавливается в конфигураторе при редактировании свойств планов счетов. Количество субконто, которое реально прикреплено к счету, не может превышать установленного максимального количества.
Метод КоличествоСубконто служит для определения количества видов субконто, прикрепленных к текущему счету в плане счетов.
Необходимо обратить внимание, что все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Эти номера выдаются в названиях колонок видов субконто в окне редактирования плана счетов; «Субконто1», «Субконто2» ... «Субконто6».
При задании видов субконто для счета их обязательно «выстраивать» по порядку один за другим: например, к счету могут быть прикреплены субконто таким образом, что их порядковые номера будут 1 и 2. Метод КоличествоСубконто возвращает общее количество видов субконто, прикрепленное к счету (в данном примере — 2), а не максимальное количество возможных видов субконто у счета.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд));
КонецЦикла;
```

ВидСубконто

Устанавливает/возвращает вид субконто счета.

Синтаксис:
```
ВидСубконто(<НомерСубконто>, <ВидСубконто>, <ТолькоОбороты>)
```
Англоязычный синоним:
```
SubcontoKind
```
Параметры:

<НомерСубконто>	Числовое выражение — порядковый номер субконто счета.
<ВидСубконто>	Необязательный параметр. Значение типа «Вид субконто», которое должно быть установлено.
<ТолькоОбороты>	Необязательный параметр. Числовое значение: 1 или 0. Данный параметр снимает или устанавливает флаг «Только обороты» для данного субкон-то счета. Если параметр не используется, то флаг «Только обороты» не меняется.

Возвращаемое значение:
Значение типа «Вид субконто».

Описание:

К любому счету или субсчету в плане счетов может быть «прикреплено» до 5 видов субконто. Максимальное количество видов субконто, которое можно «прикрепить» к счету, устанавливается в конфигураторе при редактировании свойств планов счетов. Количество субконто, которое реально прикреплено к счету, не превышает установленного максимального количества. Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Эти номера выдаются в названии колонок для видов субконто в окне редактирования плана счетов; «Субконто!», «Субконто2» ... «Субконто5».

Метод `ВидСубконто` при использовании его без второго параметра возвращает значение типа «ВидСубконто», имеющего порядковый номер, переданный в качестве первого параметра.

Метод `ВидСубконто` при вызове со вторым параметром устанавливает у счета вид субконто, имеющего порядковый номер, переданный в качестве первого параметра.

Применение данного метода следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд));
КонецЦикла;
```

ТолькоОбороты

Устанавливает/возвращает значение флага «Только обороты» субконто счета.

Синтаксис:
```
ТолькоОбороты(<НомерСубконто>, <ТолькоОбороты>]
```
Англоязычный синоним:
```
TurnoversOnly
```
Параметры:

<НомерСубконто>	Числовое выражение — порядковый номер субконто счета.
<ТолькоОбороты>	Необязательный параметр. Числовое значение: 1 или 0. Данный параметр снимает или устанавливает флаг «Только обороты» для данного субконто счета. Если параметр не используется, то флаг «Только обороты» не меняется.

Возвращаемое значение:
Значение флага «Только обороты» на момент до использования метода: 1 — флаг установлен; 0 — флаг не установлен.

Описание:

К любому счету или субсчету в плане счетов может быть «прикреплено» до 5 видов субконто. Максимальное количество видов субконто, которое можно «прикрепить» к счету, устанавливается в конфигураторе при редактировании свойств планов счетов. Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Для субконто может быть установлен флаг «Только обороты», который устанавливает для данного субконто использование только для оборотов по счету, а не для остатков.

Применение данного метода следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд) +
         ?(Сч.ТолькоОбороты(Инд) = 1, "(об.)", ""));
КонецЦикла;
```

УчетПоСумме

Устанавливает/возвращает значение флага «Учет по сумме» субконто счета.

Синтаксис:

```
УчетПоСумме(<НомерСубконто>, <УчетПоСумме>)
```

Англоязычный синоним:

```
AccountingBySum
```

Параметры:

<НомерСубконто>	Числовое выражение — порядковый номер субконто счета.
<УчетПоСумме>	Необязательный параметр. Числовое значение: 1 или 0. Данный параметр снимает или устанавливает флаг «Учет по сумме» для данного субконто счета. Если параметр не используется, то флаг «Учет по сумме» не меняется.

Возвращаемое значение:

Значение флага «Учет по сумме» на момент до использования метода: 1 — флаг установлен; 0 — флаг не установлен.

Описание:

Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Для субконто может быть установлен флаг «Учет по сумме», который устанавливает для данного субконто счета режим использования учета по сумме.

Применение данного метода для изменения флага учета следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд) +
        ? (Сч.УчетПоСумме(Инд) = 1, "(об.)", ""));
КонецЕсли;
```

УчетПоВалютнойСумме

Устанавливает/возвращает значение флага «Учет по валютной сумме» субконто счета.

Синтаксис:

```
УчетПоВалютнойСумме(<НомерСубконто>, <УчетПоСумме>)
```

Англоязычный синоним:

```
AccountingByCurrencySum
```

Параметры:

<НомерСубконто>	Числовое выражение — порядковый номер субконто счета.
<УчетПоСумме>	Необязательный параметр. Числовое значение: 1 или 0. Данный параметр снимает или устанавливает флаг «Учет по валютной сумме» для данного субконто счета. Если параметр не используется, то флаг «Учет по валютной сумме» не меняется.

Возвращаемое значение:

Значение флага «Учет по валютной сумме» на момент до использования метода: 1 — флаг установлен; 0 — флаг не установлен.

Описание:

Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Для субконто может быть установлен флаг «Учет по валютной сумме», который устанавливает для данного субконто счета режим использования учета по валютной сумме.

Применение данного метода для изменения флага учета следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить перелет итогов.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд) +
```

```
        ?(Сч.УчетПоВалютнойСумме(Инд) = 1, "(об.)", ""));
КонецЕсли;
```

УчетПоКоличеству

Устанавливает/возвращает значение флага «Учет по количеству» субконто счета.

Синтаксис:

```
УчетПоКоличеству(<НомерСубконто>, <УчетПоКоличеству>)
```

Англоязычный синоним:

```
AccountingByAmount
```

Параметры:

<НомерСубконто>	Числовое выражение — порядковый номер субконто счета.
<УчетПоКоличеству>	Необязательный параметр. Числовое значение: 1 или 0. Данный параметр снимает или устанавливает флаг «Учет по количеству» для данного субконто счета. Если параметр не используется, то флаг «Учет по количеству» не меняется.

Возвращаемое значение:

Значение флага «Учет по количеству» на момент до использования метода: 1 — флаг установлен; 0 — флаг не установлен.

Описание:

Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Для субконто может быть установлен флаг «Учет по количеству», который устанавливает для данного субконто счета режим использования учета по количеству.

Применение данного метода для изменения флага учета следует производить только в особых случаях и с учетом всех особенностей настройки учета. После изменения настроек учета система может потребовать выполнить пересчет итогов.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Для Инд = 1 По Сч.КоличествоСубконто() Цикл
   Сообщить("Субконто " + Сч.ВидСубконто(Инд) +
            ?(Сч.УчетПоКоличеству(Инд) = 1, "(об.)", ""));
КонецЕсли;
```

ИспользоватьДату

Установить дату выборки периодических реквизитов счета.

Синтаксис:

```
ИспользоватьДату(<Дата>)
```

Англоязычный синоним:

```
UseDate
```

Параметры:

<Дата>	Необязательный параметр. Выражение типа «дата».

Возвращаемое значение:

Текущее значение используемой даты (на момент до исполнения метода). **Описание:**

Метод `ИспользоватьДату` устанавливает для объекта типа «Счет» дату, на которую будут в дальнейшем выбираться (или записываться) значения периодических реквизитов счета.

Данный метод, используется обычно, если нужно обращаться сразу к не-сколькитм периодическим реквизитам спозиционированного объекта. Для одиночных обращений рекомендуется использовать методы периодического реквизита счета `Получить(<Дата>)` и `Установить(<Дата>, <Значение>)` .

Замечание: Если к объекту типа «Счет» однажды применен метод `ИспользоватьДату`, то в дальнейшем, чтобы выбирать значения периодических реквизитов, нельзя применять к этому же объекту методы `Получить` и `Установить`, т. е. в такой последовательности эти методы несовместимы.

Замечание: Метод `ИспользоватьДату` устанавливает дату выборки для данного объекта. В случае, если обращение к объекту типа счет выполняется по длинному пути (например, `Клиент.СчетУчета.СтатусУчета`) значение типа «Счет» создается динамически и после первого обращения к нему с вызовом метода `ИспользоватьДату` уничтожается, поэому при повторном обращении для получении значения реквизита данная установка уже действовать не будет.

Замечание: Метод ИспользоватьДату должен быть вызван до позиционирования объекта. В этом случае после позиционирования периодические реквизиты будут выдаваться на указанную дату.

Пример:

```
* Выведем список счетов со значением периодического реквизита "СтатусУчета".
Процедура Сформировать()
   Сч = СоздатьОбъект("Счет.Основной");
   Сч.ИспользоватьДату(РабочаяДата());
   Сч.ВыбратьСчета();
   Пока Сч.ПолучитьСчет() = 1 Цикл
      Сообщить("Статус учета " + Сч.Код + " - " + Сч.СтатусУчета);
   КонецЦикла;
КонецПроцедуры
```

ИспользоватьПланСчетов

Устанавливает план счетов, который будет использован объектом.

Синтаксис:

ИспользоватьПланСчетов(<ПланСчетов>)

Англоязычный синоним:

UseChartOfAccounts

Параметры:

<ПланСчетов> Значение типа «План Счетов».

Возвращаемое значение:

Значение типа «План Счетов». Текущее значение на момент до исполнения метода.

Описание:

Данный метод выдает и устанавливает для объекта план счетов, который будет использован им в дальнейшем для поиска, выбора, обхода счетов.

Метод ИспользоватьПланСчетов имеет смысл использовать в том случае, если объект создан функцией СоздатьОбъект для работы с разными планами счетов — СоздатьОбъект("Счет"). Если объект был создан для работы с конкретным планом счетов, то все операции поиска и обхода счетов будут работать только с этим планом счетов.

Пример:

```
Процедура Сформировать()
   Сч = СоздатьОбъект("Счет");
   Сч.ИспользоватьПланСчетов(ПланыСчетов.Основной);
   Сч.ВыбратьСчета();
   Пока Сч.ПолучитьСчет() = 1 Цикл
      Сообщить("Счет " + Сч.Код);
   КонецЦикла;
КонецПроцедуры
```

Родитель

Выдает счет вышестоящего уровня.

Синтаксис:

Родитель(<НомерУровня>)

Англоязычный синоним:

Параметры:

<НомерУровня> Необязательный параметр. Число — номер уровня. Определяет какого уровня вышестоящий счет должен быть выдан. Если параметр не указан -выдается непосредственный родитель.

Возвращаемое значение:

Значение типа «Счет». Счет вышестоящего уровня данного счета.

Описание:

При наличии в плане счетов нескольких уровней счетов-субсчетов данный метод позволяет получить счет вышестоящего уровня выбранного счета. Если параметр не задан, выдается непосредственный родитель, если задан — родитель указанного уровня.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сообщить("Родитель " + Сч.Родитель());
//Возвращает "Родитель 76"
```

ИспользоватьРодителя

Устанавливает/выдает значение родителя (счета вышестоящего уровня), используемого для выборки счетов.

Синтаксис:

```
ИспользоватьРодителя(<Родитель>, <ФлагИзменения>)
```

Англоязычный синоним:

```
UseParent
```

Параметры:

<Родитель>	Значение типа «Счет», устанавливаемое в качестве используемого родителя. Необязательный параметр. Если параметр не указан — значение используемого родителя не изменяется.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения родителя. 1 — пользователь может изменить родителя интерактивно, 0 — пользователь не может интерактивно изменить родителя.

Возвращаемое значение:

Значение типа «Счет». Текущее значение используемого родителя на момент до исполнения метода.

Описание:

При наличии в плане счетов нескольких уровней счетов-субсчетов при переборе или выборе счета существует возможность отбирать только счета подчиненные некоторому счету. Метод ИспользоватьРодителя устанавливает значение, которе будет затем использовано методами перебора и выбора счета.

Пример:

```
Процедура Сформировать ()
   Сч = СоздатьОбъект("Счет.Основной");
   Сч.ИспользоватьРодителя(СчетПоКоду("01"));
   Сч.ВыбратьСчета();
   Пока Сч.ПолучитьСчет(0) = 1 Цикл
      Сообщить(Сч.Код);
   КонецЦикла;
КонецПроцедуры
```

КодСубсчета

Выдает код счета без кодов счетов вышестоящих уровней.

Синтаксис:

```
КодСубсчета()
```

Англоязычный синоним:

```
SubAccountCode
```

Возвращаемое значение:

Строковое значение — код счета без кодов счетов вышестоящих уровней.

Описание:

При наличии в плане счетов нескольких уровней счетов-субсчетов атрибут "Код" объекта типа «Счет» содержит полный код счета с учетом всех вышестоящих счетов через разделитель (точку). Метод КодСубсчета выдает собственно код даного счета без кодов счетов вышестоящих уровней.

Пример:

```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сообщить("Код счета " + Сч.Код); //Возвращает "76.02"
Сообщить("Код Субсчета " + Сч.КодСубсчета); //Возвращает "76"
```

Блокировка

Установить/прочитать режим блокировки.

Синтаксис:

```
Блокировка(<ВклВыкл>)
```

Англоязычный синоним:

```
Locking
```

Параметры:

<ВклВыкл>	Необязательный параметр. Число: 1 — включить; 0 — выключить.

Возвращаемое значение:

Если при вызове метода параметр <ВклВыкл> не задан, то возвращается Режим блокировки до выполнения метода. Число: 1 — заблокирован; 0 — свободен.

Если при вызове метода параметр <ВклВыкл> задан, то возвращается результат выполнения метода блокировки. Число: 1 — успешно; 0 — не получилось.

Описание:
Метод Блокировка позволяет установить/прочитать режим блокировки.
Пример:
```
Блокировка(0);
```

Новый

Создает новый счет.
Синтаксис:
```
Новый(<ЕстьСубсчета>)
```
Англоязычный синоним:
```
New
```
Параметры:

<ЕстьСубсчета>	Числовое выражение — признак наличия субсчетов у создаваемого счета. Может принимать значения: 0 — счет не будет иметь субсчета; 1 — счет будет иметь субсчета. По умолчанию 0;

Описание:
Создает новый счет. При добавлении нового счета необходимо указать, будет ли счет иметь субсчета. То есть будет он являться собственно счетом или группой.

Если счет будет иметь субсчета, то такой счет нельзя будет указать в качестве корреспондирующего счета при вводе проводок — необходимо будет указывать субсчета этого счета. Иначе, если счет не будет иметь субсчетов, его можно указывать при вводе проводок.

Однако, если для счета, для которого было указано отсутствие субсчетов, все-таки введен субсчет, то система 1С:Предприятие выполнит следующее:

- Будет создан новый счет (группа) с тем же кодом;
- Код данного счета будет дополнен субсчетом 0;

В дальнейшем код субсчета 0 можно будет изменить на другой.

Замечание. Метод Новый сам по себе не выполняет запись нового счета в информационную базу, для внесения нового счета в список счетов, следует выполнить метод Записать.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.Новый();
Сч.Код = "76.02";
Сч.Записать();
```

НазначитьТип

Назначить тип для реквизита неопределенного вида.
Синтаксис:
```
НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>)
```
Англоязычный синоним:
```
SetType
```
Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита счета неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту счета. Например: "Строка", "Число", "Справочник.Товары", "Документ. РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных. Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:
Метод НазначитьТип позволяет назначить тип для реквизита, которому в конфигураторе назначен тип «Неопределенный».
Пример:
```
ВыбСчет.НазначитьТип("ТМЦ", "Справочник.Товары");
```

Записать

Записывает изменения счета.
Синтаксис:
```
Записать()
```
Англоязычный синоним:
```
Write
```
Описание:
Метод `Записать` выполняет запись (обновление) счета. Все изменения сделанные в существующем счете, или созданном счете будут занесены в информационную базу только после вызова метода Записать.

Замечание. Если этот метод применяется в Модуле формы счета непосредственно к счету локального контекста, то данный метод отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Записать".

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сч.Наименование = "Прочие взаиморасчеты";
Сч.Записать();
```

Удалить

Удаляет счет из плана счетов.
Синтаксис:
```
Удалить(<Режим>)
```
Англоязычный синоним:
```
Delete
```
Параметры:
`<Режим>` Числовое выражение: 1 — непосредственное удаление; 0 — пометка на удаление. По умолчанию 1;
Описание:
Метод `Удалить` удаляет (или помечает на удаление) текущий счет. Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.

Замечание. Непосредственное удаление объекта следует применять только в особых случаях, так как счет может участвовать в проводках и непосредственное удаление может повлечет нарушение в бухгалтерском учете.

Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сч.Удалить(0);
```

СнятьПометкуУдаления

Снять пометку удаления счета.
Синтаксис:
```
СнятьПометкуУдаления()
```
Англоязычный синоним:
```
ClearDeleteMark
```
Описание:
Метод `СнятьПометкуУдаления` снимает пометку удаления текущего счета.
Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.
Пример:
```
Сч = СоздатьОбъект("Счет.Основной");
Сч.НайтиПоКоду("76.02");
Сч.СнятьПометкуУдаления();
```

Методы контекста Модуля формы списка счетов

Описанные в данном разделе методы доступны только в контексте Модуля формы списка счетов (см. «Виды программных модулей»).

ИспользоватьДату

Установить дату выборки периодических реквизитов формы списка счетов.

Синтаксис:

```
ИспользоватьДату(<Дата>)
```

Англоязычный синоним:

```
UseDate
```

Параметры:

<Дата>	Выражение со значением типа «дата»

Возвращаемое значение:

Значение даты выборки периодических реквизитов формы списка счетов.

Описание:

Метод ИспользоватьДату устанавливает для формы списка счетов дату, на которую будут в дальнейшем выбираться (или записываться) значения периодических реквизитов справочника.

Данный метод доступен только в контексте Модуля формы списка счетов (см. «Виды программных модулей»). Действие данного метода относится только к текущему справочнику, который доступен в локальном контексте Модуля формы списка счетов.

Пример:

```
ИспользоватьДату(РабочаяДата());
```

ИспользоватьПланСчетов

Установить (получить) текущий план счетов формы списка счетов.

Синтаксис:

```
ИспользоватьПланСчетов(<ПланСчетов>)
```

Англоязычный синоним:

```
UseChartOfAccounts
```

Параметры:

<ПланСчетов>	Значение типа «ПланСчетов».

Возвращаемое значение:

Значение текущего плана счетов формы до вызова метода.

Описание:

Метод ИспользоватьПланСчетов устанавливает текущий план счетов в форме списка счетов, если форма открыта для просмотра нескольких планов счетов. Если параметр не задан, то метод позволяет получить текущий план счетов.

Данный метод доступен в контексте Модуля формы списка счетов (см. «Виды программных модулей»).

Пример:

```
Процедура Печать()
   Сч = СоздатьОбъект("Счет");
   Сч.ИспользоватьПланСчетов(ИспользоватьПланСчетов());
КонецПроцедуры;
```

ИспользоватьРодителя

Установить родителя для формы списка счетов.

Синтаксис:

```
ИспользоватьРодителя(<Родитель>, <ФлагИзменения>)
```

Англоязычный синоним:

```
UseParent
```

Параметры:

<Родитель>	Выражение со значением счета.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения родителя. 1 — пользователь может изменить родителя интерактивно, 0 — пользователь не может интерактивно изменить родителя.

Возвращаемое значение:

Значение родителя для формы списка счетов (до применения метода).

Описание:

Метод ИспользоватьРодителя устанавливает счет в качестве основного для формы списка счетов (показываются субсчета выбранного счета).

Данный метод доступен в контексте Модуля формы списка счетов (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку счетов, который доступен в локальном контексте Модуля формы списка счетов.

При добавлении нового субсчета текущего плана счетов данный параметр также будет являться свойством нового субсчета.

Пример:
```
ИспользоватьРодителя(ВыбСчет, 1);
```

ИспользоватьКорСчет

Установить счет для показа списка корреспондирующих счетов.

Синтаксис:
```
ИспользоватьКорСчет(<Счет>, <Корреспонденция>)
```
Англоязычный синоним:
```
UseCorAccount
```
Параметры:

<Счет>	Выражение со значением счета.
<Корреспонденция>	Число: 1 — дебетовый счет; 0 — кредитовый счет.

Описание:

Метод ИспользоватьКорСчет устанавливает счет для показа списка корреспондирующих с ним счетов. Корреспондирующие счета задаются с помощью корректных проводок.

Данный метод доступен в контексте Модуля формы списка счетов (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку счетов, который доступен в локальном контексте Модуля формы списка счетов.

Пример:
```
ИспользоватьКорСчет(ВыбСчет, 1);
```

ИерархическийСписок

Установить режим иерархического списка счетов.

Синтаксис:
```
ИерархическийСписок(<ФлагИерархСписка>, <ФлагИзменения>)
```
Англоязычный синоним:
```
HierarchicalList
```
Параметры:

<ФлагИерархСписка>	Флаг иерархического списка. 1 — иерархический список; 0 — неиерархический список.
<ФлагИзменения>	Необязательный параметр. Этим флагом регулируется возможность интерактивного изменения флага иерархического списка. 1 — пользователь может изменить иерархичность интерактивно, 0 — пользователь не может интерактивно изменить иерархичность.

Возвращаемое значение:

Значение флага иерархического списка для формы списка счетов (до применения метода).

Описание:

Метод ИерархическийСписок устанавливает режим иерархического списка счетов.

Данный метод доступен в контексте Модуля формы списка счетов (см. «Виды программных модулей»). Действие данного метода относится ко всему текущему списку счетов, который доступен в локальном контексте Модуля формы списка счетов.

Пример:
```
ИерархическийСписок(1, 1);
```

РедактироватьВДиалоге

Установить способ редактирования счета.

Синтаксис:
```
РедактироватьВДиалоге(<Способ>, <Разрешение>)
```
Англоязычный синоним:
```
EditInForm
```
Параметры:

<Способ>	Необязательный параметр. Способ редактирования счета: 1 — в диалоге; 0 — в строке.
<Разрешение>	Необязательный параметр. Флаг разрешения пользователю менять способ редактирования: 1 —разрешить; 0 — запретить.

Возвращаемое значение:

Значение установленного на данный момент способа счета (до применения метода).

Описание:

Метод РедактироватьВДиалоге устанавливает способ редактирования счета.

Данный метод доступен в контексте Модуля формы списка счетов. Действие данного метода относится ко всему текущему списку счетов, который доступен в локальном контексте Модуля формы списка.

Пример:
```
ТекСп = РедактироватьВДиалоге();
// какой сейчас способ?
РедактироватьВДиалоге(1);
// установить редактирование в диалоге
РедактироватьВДиалоге(1, 0);
// установить редактирование в диалоге и запретить его менять
```

ВыборГруппы

Устанавливает или сбрасывает признак выбора группы.

Синтаксис:
```
ВыборГруппы([<ФлагВыбораГруппы>])
```
Англоязычный синоним:
```
SelectGroup
```
Параметры:

<ФлагВыбораГруппы>	Необязательный параметр. Числовое выражение: 1 — выбирать группы; 0 — не выбирать группы.

Возвращаемое значение:
Текущее числовое значение режима выборки групп (на момент до исполнения метода).

Описание:

Метод ВыборГруппы устанавливает режим выборки групп для формы списка счетов, которая открыта в режиме выбора или подбора счета. Открытие такой формы списка счетов обычно производится вызовом метода Выбрать или нажатием кнопки выбора, если выбираемый счет является элементом диалога.

По умолчанию выбор группы разрешен. Режим, устанавливаемый данным методом для списка счетов, влияет на интерактивный выбор пользователем значения типа «Счет» из списка счетов (плана счетов). Установка режима позволяет разрешить или запретить пользователю выбирать счета-группы (счета, которые могут иметь субсчета).

Пример:
```
ВыборГруппы(0);
```

Предопределенные процедуры модуля формы списка счетов

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях форм списка счетов (см. «Виды программных модулей»).

В основном, данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ПриВводеСтроки

Предопределенная процедура при вводе новой строки списка счетов

Синтаксис:
```
ПриВводеСтроки()
```
Англоязычный синоним:
```
OnNewLine
```
Описание:

Вызов предопределенной процедуры ПриВводеСтроки производится в системе 1С:Предприятие при интерактивном вводе новой строки (до начала ввода) в форме списка счетов. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя вводить новые строки списка счетов), то новая строка списка счетов не будет инициирована.

Данная предопределенная процедура может располагаться в Модуле формы списка счетов (см. «Виды программных модулей»).

Пример:
```
Процедура ПриВводеСтроки()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права добавлять счета!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриРедактированииНовойСтроки

Предопределенная процедура при редактировании новой строки списка счетов.

Синтаксис:

ПриРедактированииНовойСтроки()

Англоязычный синоним:

OnEditNewLine

Описание:

Вызов предопределенной процедуры ПриРедактированииНовойСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования новой строки списка счетов (после того, как новая строка уже заведена). Данная процедура может использоваться, например, для установки начальных значений (значений по умолчанию) характеристик нового счета. В данной предопределенной процедуре установка статуса возврата не имеет смысла, т. к. отказаться от ввода новой строки в этот момент уже невозможно.

Данная предопределенная процедура может располагаться в Модуле формы списка счетов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриРедактированииНовойСтроки()
   СтатусСчета = ОснСтатусСчетов;
КонецПроцедуры
```

ПриНачалеРедактированияСтроки

Предопределенная процедура при начале редактирования существующей строки списка счетов.

Синтаксис:

ПриНачалеРедактированияСтроки()

Англоязычный синоним:

OnStartEditLine

Описание:

Вызов предопределенной процедуры ПриНачалеРедактированияСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования существующей строки списка счетов (кроме новой). Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя изменять характеристики бухгалтерских счетов), строка не будет изменена.

Данная предопределенная процедура может располагаться в Модуле формы списка счетов (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права менять счета!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыбореРодителя

Предопределенная процедура выбора родительской группы счета.

Синтаксис:

ПриВыбореРодителя(<Счет>)

Англоязычный синоним:

OnSetParent

Параметры:

<Счет>	Значение счета, который интерактивно установлен в качестве родителя.

Описание:

Вызов предопределенной процедуры ПриВыборе Родителя производится в системе 1С:Предприятие при интерактивной смене родительской группы счета (выбор следующего или предыдущего уровня). Если в данной предопределенной процедуре установить статус возврата — 0, то выбор родительской группы не будет произведен.

Данная предопределенная процедура может располагаться только в Модуле формы списка счетов.

Пример:

```
Процедура ПриВыбореРодителя(Родитель)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если Родитель = ЗапрещеннаяГруппа Тогда
         Предупреждение("Вам запрещено изменять счет!", 2);
```

```
            СтатусВозврата(0);
         КонецЕсли;
      КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура при записи строки списка счетов.

Синтаксис:

ПриЗаписи(<СписокПериодРекв>)

Англоязычный синоним:

OnWrite

Параметры:

<СписокПериодРекв>	Строковое значение — список разделенных запятыми изменяемых периодических реквизитов счета. В данный параметр система 1С:Предприятие передает перечень периодических реквизитов, которые были интерактивно выбраны пользователем для обновления в окне диалога выбора.

Описание:

Вызов предопределенной процедуры ПриЗаписи производится системой 1С:Предприятие при интерактивной записи строки списка счетов. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя изменять некоторые характеристики счета), то запись строки списка счетов не будет выполнена.

Формальный параметр <СписокПериодРекв> используется в теле процедуры для обработки ситуаций изменений периодических реквизитов счета.

Данная предопределенная процедура может располагаться в следующих программных модулях: модуль формы счета, модуль формы списка счетов, (см. «Виды программных модулей»).

Пример:

```
Процедура ПриЗаписи(СписокРекв)
   Если Валютный = 1 Тогда
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Предопределенные процедуры модуля формы счета

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в модуле формы счета (см. «Виды программных модулей»).

В основном, данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ВводНового

Предопределенная процедура при вводе нового счета.

Синтаксис:

ВводНового(<ПризнКопирования>, <ОбъектКопирования>)

Англоязычный синоним:

InputNew

Параметры:

<ПризнКопирования>	Признак того, что объект введен копированием. Число: 1 — объект введен копированием, 0 — просто новый объект. Данный признак может быть использован для анализа необходимости инициализации реквизитов нового объекта.
<ОбъектКопирования>	Объект, который был скопирован.

Описание:

Вызов процедуры ВводНового производится в системе 1С:Предприятие неявно в момент выбора открытия формы счета для ввода нового счета. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) характеристик нового счета.

Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя вводить новые счета), ввода нового счета и открытие формы счета не будет выполнено.

Процедура ВводНового контекста справочников должна находиться в модуле: формы счета (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНового()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права добавлять счета!", 2);
      СтатусВозврата(0);
   КонецЕсли;
   СтатусСчета = ОснСтатусСчетов;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура при записи счета.

Синтаксис:

ПриЗаписи(<СписокПериодРекв>)

Англоязычный синоним:

OnWrite

Параметры:

<СписокПериодРекв>	Строковое значение — список разделенных запятыми изменяемых периодических реквизитов счета. В данный параметр система 1С:Предприятие передает перечень периодических реквизитов, которые были интерактивно выбраны пользователем для обновления в окне диалога выбора.

Описание:

Вызов предопределенной процедуры ПриЗаписи производится системой 1С:Предприятие при интерактивной записи счета. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя изменять некоторые характеристики счета), то запись счета не будет выполнена.

Формальный параметр <СписокПериодРекв> используется в теле процедуры для обработки ситуаций с изменениями периодических реквизитов счета.

Процедура ПриЗаписи контекста справочников должна находиться в модуле формы счета (см. «Виды программных модулей»).

Пример:

```
Процедура ПриЗаписи(СписокРекв)
   Если Валютный = 1 Тогда
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 17
Работа с операциями и проводками

Для отражения в бухгалтерском учете информации о движении средств в системе 1С:Предприятие используются `Операции` и `Проводки`. Эти объекты поддерживаются компонентой «Бухгалтерский учет». Настройка свойств операций и проводок производится соответствующими объектами метаданных в конфигурации. В руководстве по конфигурированию можно подробно ознакомиться с основными принципами и свойствами операций и проводок.

Контекст работы с операциями и проводками

Для манипулирования данными операций и проводок из встроенного языка 1С:Предприятия используется агрегатный объект типа «Операция». Так как проводки в системе 1С:Предприятие принадлежат операциям, то управление и операциями и проводками выполняется объектом «Операция».

Во-первых, объект «Операция» используется для формирования и анализа проводок формируемых документом. Для этого у агрегатного объекта «Документ» существует атрибут `Операция` типа «Операция», который обеспечивает доступ к операции данного документа.

Во-вторых, объект «Операция» непосредственно доступен в контекстах формы операции, формы журнала операции и формы журнала проводок. То есть в модулях указанных форм обращение к атрибутам и методам текущего объекта выполняется напрямую. В форме журнала операций текущим объектом является операция, на которой в данный момент стоит курсор в форме журнала. В форме журнала проводок текущим объектом является операция, которой принадлежит проводка, на которой в данный момент стоит курсор в форме журнала.

В-третьих, объект «Операция» используется для перебора существующих операций и проводок при формировании отчетов и других выборок. В этом случае объект создается при помощи вызова функции `СоздатьОбъект ("Операция")`. Англоязычный синоним ключевого слова `Операция` — `Operation`.

В этом варианте использования становятся доступны методы перебора операций и проводок различных операций, а также добавления и удаления операции.

Метод записи операции доступен также при проведении документа.

Как уже отмечалось, доступ к проводкам также осуществляется объектом «Операция». Методы и атрибуты объекта «Операция» осуществляют доступ к данным как собственно операции, так и к данным проводок. Операция содержит набор методов для перебора и позиционирования на конкретные проводки. При этом одна из проводок операции может быть текущей. Все обращения к атрибутам и методам данных проводки будут относится именно к текущей проводке. Если текущей проводки нет, то обращение к атрибутам и методам проводки недоступно.

Одной из возможностей организации данных бухгалтерского учета системы 1С:Предприятие является поддержка сложных проводок. Под сложной проводкой понимается проводка, состоящая из нескольких корреспонденции (одного дебета — нескольких кредитов или наоборот). С точки зрения синтаксиса объекта «Операция» текущей проводкой может являться простая проводка или одна из корреспонденции сложной проводки. Нумерация проводок осуществляется номером проводки и номером корреспонденции.

С точки зрения чтения, все операции используются практически одинаково. Однако изменения (создание новых, запись, удаление) операций сильно отличается для операций двух типов.

«Ручные» операции — это операции, введенные пользователем вручную или записанные объектом типа «Операции», созданным функцией `СоздатьОбъект`. Все изменения из встроенного языка этих операций производятся объектами типа «Операции», созданными функцией `СоздатьОбъект`. Такие операции принадлежат документам специального вида «Операция», который не имеет большинства свойств обычных документов и используется только для отражении в журналах документов «Ручных» операций. Вид документа «Операция» создается системой автоматически при внесении в конфигурацию первого объекта метаданных типа «План счетов».

Операции принадлежащие документам — это операции, которые создаются документами различных видов, которым установлен признак «Бухгалтерский учет» (кроме вида «Операция»). Эти операции могут записываться только при записи документов или при проведении документов. Наличие операции у кон-кертного документа регламентируется в метаданных в настройках вида документа. Операция для документа конкретного вида может создаваться либо при записи (режим «Всегда»), либо создаваться выборочно (режим «Выборочно») (при помощи метода документа `СуществуетОперация`), либо только при проведении документа (режим «только при проведении»). В первых двух случаях изменение реквизитов самой операции может производиться в модуле формы документа и при записи они автоматически будут записаны в операцию принадлежащую документу. Однако проводки в операцию принадлежащую документу могут быть записаны в любом случае только в процессе проведения документа (предопределенной процедуре `ОбработкаПроведения`). При этом в процессе проведения документа для записи операции следует вызвать метод операции `Записать`. Операция созданная документом в процессе проведения при перепроведении документа сначала уничтожается, а при выполнении процедуры `ОбработкаПроведения` записывается заново. Если документ делается не проведенным, то проводки из операции удаляются, а сама операция уничтожается, если она была записана в режиме «только при проведении».

Атрибуты объекта «Операция»

ДатаОперации

Дата операции.
Синтаксис:
ДатаОперации
Англоязычный синоним:
OperDate
Значение:
Значение типа «Дата».
Описание:
Атрибут ДатаОперации содержит дату бухгалтерской операции. Так как каждая бухгалтерская операция принадлежит документу (причем только одному, а операция введенная вручную также принадлежит специальному документу «Операция»), значение атрибута совпадает со значением атрибута ДатаДок документа, которому принедлежит операция. Наличие данного атрибута у оп-реации упрощает обращение к дате документа, которому принадлежит операция.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Опер.ДатаОперации);
```

Содержание

Содержание операции.
Синтаксис:
Содержание
Англоязычный синоним:
Description
Значение:
Значение типа «Строка».
Описание:
Атрибут Содержание служит для обращения к содержанию операции. Содержание операции представляет собой произвольную символьную строку. Максимальная длина содержания задается в конфигураторе при редактировании свойств объекта метаданных «Операция».
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Опер.Содержание);
```

СуммаОперации

Сумма операции.
Синтаксис:
СуммаОперации
Англоязычный синоним:
OperSum
Значение:
Значение типа «Число».
Описание:
Атрибут СуммаОперации служит для обращения к сумме операции. Сумма операции является положительным или отрицательным числом. Сумма операции предназначена лишь для иллюстрирования денежного выражения операции и не влияет на бухгалтерские итоги. Длина и точность суммы операции задается конфигураторе при редактировании свойств объекта метаданных «Операция».
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Опер.СуммаОперации);
```

<РеквизитОперации>

Возвращает или задает значение дополнительного реквизита операции.
Синтаксис:
<РеквизитОперации> Идентификатор дополнительного реквизита операции, как он задан в конфигураторе.
Значение:

Тип значения определяется типом реквизита заданным в метаданных.

Описание:

Помимо реквизитов операции, задаваемых на системном уровне (сумма, содержание), в процессе конфигурирования для операции можно создать практически неограниченное число дополнительных реквизитов для хранения любой необходимой информации.

Атрибут <Реквизит> позволяет обращаться к значению дополнительного реквизита операции. Для обращения к конкретному реквизиту должен быть указан его идентификатор, заданный для этого реквизита в конфигураторе.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Опер.ПризнакУчета);
```

Документ

Документ, которому принадлежит операция.

Синтаксис:

```
Документ
```

Англоязычный синоним:

```
Document
```

Значение:

Значение типа документ.

Описание:

Атрибут Документ является атрибутом «только для чтения», то есть ему нельзя присваивать никакого значения. Он служит для обращения к документу, котрому принадлежит операция. Так, как любая операция принадлежит документу (операция введенная «вручную» принадлежит документу специального вида «Операция») доступ к некоторым свойствам операции, например, номеру возможен только посредством обращения к атрибуту Документ.

Атрибут Документ предназначен только для доступа к атрибутам и реквизитам документа, которому принадлежит операция. Для получения значения типа «Документ» используется метод ТекущийДокумент.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Опер.Документ.НомерДок);
```

Сумма

Сумма выбранной проводки (корреспонденции) операции.

Синтаксис:

```
Сумма
```

Англоязычный синоним:

```
Sum
```

Значение:

Значение типа «Число».

Описание:

Атрибут Сумма служит для обращения к сумме выбранной проводки (корреспонденции) операции.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Сумма " + Опер.Сумма);
КонецЦикла;
```

Валюта

Валюта выбранной проводки или корреспонденции операции.

Синтаксис:

```
Валюта
```

Англоязычный синоним:

```
Currency
```

Значение:

Значение типа «Справочник», вид которого определен в настройке валютного учета в конфигурации.

Описание:

Атрибут Валюта служит для обращения к валюте выбранной проводки или корреспонденции операции.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДоку мент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Валюта " + Опер.Валюта + " Вал.сумма=" + Опер.ВалСумма);
КонецЦикла;
```

ВалСумма

Валютная сумма выбранной проводки или корреспонденции операции.

Синтаксис:

ВалСумма

Англоязычный синоним:

CurSum

Значение:

Значение типа «Число».

Описание:

Атрибут ВалСумма служит для обращения к сумме в валюте выбранной проводки или корреспонденции операции.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Валюта " + Опер.Валюта + " Вал.сумма=" + Опер.ВалСумма);
КонецЦикла;
```

Количество

Количество выбранной проводки или корреспонденции операции.

Синтаксис:

Количество

Англоязычный синоним:

Amount

Значение:

Значение типа «Число».

Описание:

Атрибут Количество служит для обращению к количеству выбранной проводки или корреспонденции операции.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Количество ="+0пер.Количество);
КонецЦикла;
```

<РеквизитПроводки>

Обращение к дополнительному реквизиту проводки.

Синтаксис:

<РеквизитПроводки> Идентификатор дополнительного реквизита проводки, как он задан в конфигураторе.

Описание:

Помимо реквизитов проводки, задаваемых на системном уровне (сумма, валюта и других), в процессе конфигурирования для проводки можно создать практически неограниченное число дополнительных реквизитов для хранения любой необходимой информации.

Атрибут <РеквизитПроводки> позволяет обращаться к значениям дополнительных реквизитов проводки. Для обращения к конкретному реквизиту должен быть указан его идентификатор, заданный для этого реквизита в конфигураторе.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Количество =" + Опер.Количество);
КонецЦикла;
```

Дебет

Обращение к дебету проводки (корреспонденции).

Синтаксис:

```
Дебет
```

Англоязычный синоним:

```
Debit
```

Описание:

Атрибут Дебет служит для обращения к дебетовой части текущей проводки (корреспонденции) операции. Данный атрибут возвращает агрегатный объект специального типа «Дебет», который используется только для доступа к данным дебетовой части конкретной проводки (корреспонденции) операции и не используется отдельно.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.Дебет.Счет + " - " + Опер.Кредит.Счет);
КонецЦикла;
```

Кредит

Обращение к кредиту проводки (корреспонденции).

Синтаксис:

```
Кредит
```

Англоязычный синоним:

```
Credit
```

Описание:

Атрибут Кредит служит для обращения к кредитовой части текущей проводки (корреспонденции) операции. Данный атрибут возвращает агрегатный объект специального типа «Кредит», который используется только для доступа к данным кредитовой части конкретной проводки (корреспонденции) операции и не используется отдельно.

Перед обращением к атрибуту конкретной проводки эта проводка должна быть выбрана с помощью методов ВыбратьПроводки, ПолучитьПроводку или создана при помощи метода НоваяПроводка. В модулях форм «Операции» и «Журнал проводок» текущая проводка определяется положением курсора.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.Дебет.Счет + " - " + Опер.Кредит.Счет);
КонецЦикла;
```

Атрибуты объектов «Дебет» и «Кредит»

Счет

Счет дебета/кредита проводки (корреспонденции) операции.
Синтаксис:
Счет
Англоязычный синоним:
Account
Значение:
Значение типа «Счет».
Описание:
Атрибут Счет служит для доступа к счету дебета/кредита проводки или корреспонденции операции.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.Дебет.Счет.Наименование + " - " +
                          Опер.Кредит.Счет.Наименование);
КонецЦикла;
```

<Субконто>

Субконто дебета/кредита проводки (корреспонденции) операции.
Синтаксис:

<Субконто>	Идентификатор вида субконто, как он задан в конфигураторе.

Описание:
Атрибут <Субконто> служит для доступа к значению субконто дебета/кредита проводки (корреспонденции) операции. В конкретном случае обращение к субконто для дебета/кредита определяется счетом дебета/кредита, то есть заданными для него видами субконто по которым ведется аналитический учет по этому счету.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Если Опер.Дебет.Счет=СчетПоКоду("60") Тогда
      Сообщить("Субконто " + Опер.Дебет.Организация);
   КонецЕсли;
КонецЦикла;
```

Методы объектов «Дебет» и «Кредит»

Субконто

Обращение к субконто дебета/кредита проводки (корреспонденции) операции.
Синтаксис:
Субконто(<ПорядковыйНомерСубконто> | <ВидСубконто>, <Субконто>)
Англоязычный синоним:
Subconto
Параметры:

<ПорядковыйНомерСубконто>	Необязательный параметр. Числовое выражение — порядковый номер субконто. По умолчанию 1.
<ВидСубконто>	Значение типа «Вид субконто».
<Субконто>	Необязательный параметр. Значение субконто. Если параметр не задан — значение не изменяется.

Возвращаемое значение:
Если указан порядковый номер вида субконто или вид субконто, метод возвращает значение субконто.
Описание:
К любому счету или субсчету в плане счетов может быть «прикреплено» до 5 видов субконто. Максимальное количество видов субконто, которое можно «прикрепить» к счету, устанавливается в конфигураторе при редактировании свойств

планов счетов. Количество субконто, которое реально прикреплено к счету, не превышает установленного максимального количества.

Все виды субконто, прикрепленные к счету в плане счетов, имеют порядковые номера. Эти номера выдаются в названии колонок для видов субконто в окне редактирования плана счетов; «Субконто1», «Субконто2» ... «Субконто5».

Метод Субконто дебета/кредита проводки служит для получения и установки значения субконто по его номеру или виду соотвественно в дебете или в кредите проводки (коррепспонденции) операции.

Возможность использования субконто конкретного вида или номера определяется счетом дебета/кредита проводки.

Если при использовании метода Субконто первым параметром передается порядковый номер субконто или вид субконто, то метод возвратит значение субконто, имеющего этот номер.

Если вторым параметром передается значение субконто, метод присваивает указаное значение виду субконто проводки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Для Инд = 1 По Опер.Дебет.Счет.КоличествоСубконто() Цикл
      Сообщить("Субконто " + Опер.Дебет.Субконто(Инд));
   КонецЦикла;
КонецЦикла;
```

ПредставлениеСубконто

Представление субконто дебета/кредита проводки (корреспонденции) операции.

Синтаксис:

```
ПредставлениеСубконто(<ПорядковыйНомерСубконто> | <ВидСубконто>, <Режим>)
```

Англоязычный синоним:

```
SubcontoPresentation
```

Параметры:

<ПорядковыйНомерСубконто>	Необязательный параметр. Числовое выражение — порядковый номер субконто. По умолчанию 1.
<ВидСубконто>	Идентификатор вида субконто.
<Режим>	Необязательный параметр. Числовое выражение — признак полноты выдачи представления субконто. Может принимать значения: 0 — полное представление; 1 — краткое представление. По умолчанию 0;

Возвращаемое значение:

Символьная строка — представление субконто.

Описание:

Представлением называется символьная строка, содержащая информацию из реквизитов субконто. Эта символьная строка может быть использована для отображения значений реквизитов субконто в различных отчетах, диалогах и других визуальных элементах конфигурации.

Представление может быть настроено в конфигурации для видов субконто типа «Справочник» или «Документ». Формат представления определяется в конфигураторе при редактировании свойств вида субконто.

Метод ПредставлениеСубконто позволяет получить представление для субконто дебета/кредита текущей проводки (корреспонденции).

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Для Инд = 1 По Опер.Дебет.Счет.КоличествоСубконто() Цикл
      Сообщить("Субконто " + Опер.Дебет.ПредставлениеСубконто(Инд));
   КонецЦикла;
КонецЦикла;
```

Методы объекта «Операция»

ВыбратьОперации

Открывает выборку операций за период.

Синтаксис:

```
ВыбратьОперации(<НачалоПериода>, <КонецПериода>)
```

Англоязычный синоним:
SelectOpers
Параметры:

<НачалоПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки операций.
<КонецПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки операций.

Возвращаемое значение:
Число: 1 — действие выполнено и в выборке есть хотя бы одна операция;
0 — действие не выполнено или в выборке нет ни одной операции.
Описание:
Метод ВыбратьОперации открывает выборку, содержащую операции за указанный период.
Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Сообщить("Операция " + Опер.Содержание);
КонецЦикла;
```

ВыбратьОперацииСПроводками

Открывает выборку операций с проводками за указанный период.
Синтаксис 1:

```
ВыбратьОперацииСПроводками(<НачалоПериода>, <КонецПериода>, <Фильтр>, <Валюта>,
                           <ПланСчетов>, <РазделительУчета>)
```

Синтаксис 2:

```
ВыбратьОперацииСПроводками(<НачалоПериода>, <КонецПериода>, <Счет>, <КорСчет>,
                           <Флаг>, <Валюта>, <ПланСчетов>, <РазделительУчета>)
```

Англоязычный синоним:
SelectOpersAndEntries
Параметры:

<НачалоПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки операций.
<КонецПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки операций.
<Фильтр>	Необязательный параметр. Строка — условие отбора проводок.
<ПланСчетов>	Необязательный параметр. Значение типа «План Счетов». Если параметр не указан — по всем планам счетов.
<РазделительУчета>	Необязательный параметр. Значение разделителя учета. Если параметр не указан — по всем значениям разделителя учета.
<Счет>	Необязательный параметр. Счет — счет, по которому будут отбираться проводки.
<КорСчет>	Необязательный параметр. Счет — корреспондирующий счет, по которому будут отбираться проводки. Параметр имеет смысл, если указан параметр <Счет>.
<Флаг>	Необязательный параметр. Число — признак вида оборота. Параметр может принимать значения: 1 — отбирать проводки только по дебету счета; 2 — отбирать проводки только по кредиту счета; 3 — отбирать проводки и по дебету, и по кредиту. По умолчанию 3.
<Валюта>	Необязательный параметр. Значание типа «справочник» (вида справочника используемого для валютного учета) — признак отбора проводок по валюте.

Возвращаемое значение:
Число: 1 — действие выполнено и в выборке есть хотя бы одна проводка;
0 — действие не выполнено или в выборке нет ни одной проводки.
Описание:
Метод ВыбратьОперацииСПроводками служит для выбора проводок операций за период. Параметрами могут являться различные условия отбора проводок.

После вызова данного метода метод ПолучитьПроводку выбирает очередную проводку из выборки удовлетворяющую заданным условиям, а метод ПолучитьОперацию выбирает первую проводку слудующей операции удовлетворяющую заданным условиям.

Метод может вызываться с двумя вариантами набора параметров. В первом варианте основные условия отбора задаются специальной строкой-фильтром, во втором указанием счета и корреспондирующего счета.

В параметре <Фильтр> задаются критерии отбора проводок для включения в выборку. Если параметр не заполнен, в выборку включаются все проводки. В общем случае в параметре <Фильтр> могут находиться одна или несколько корреспонденции счетов или символьных строк, разделяемых точкой с запятой «;». Символьные строки представляют собой наборы символов, заключенные в кавычки (при передаче строки в явном виде в параметре внутри строки двойные кавычки задаются двумя символами двойных кавычек). Корреспонденции имеют вид:

n — проводки со счетом n;

n, m — проводки в дебет счета n с кредита счета m.

Здесь в качестве n и m может указываться звездочка («*»), она обозначает любой счет. Например, *, 51 — все проводки с кредита 51 счета.

В выборку включаются все проводки, удовлетворяющие следующим условиям:

Если в параметре <Фильтр> указаны корреспонденции счетов, то проводка должна соответствовать одной из этих корреспонденции.

Если параметре <Фильтр> указаны строки символов, то в проводке должна содержаться хотя бы одна из этих строк — либо в содержании операции, либо представлениях значений субконто и реквизитов проводки и операции.

Примеры:

50	— все проводки со счетом 50;
50, *	— все проводки в дебет 50 счета;
* , 51	— все проводки с кредита 51 счета;
50, 51	— все проводки в дебет 50 счета с кредита 51;
51; 52	— все проводки со счетом 51 или счетом 52;
46, 68.2	— все проводки в дебет 46 счета с кредита 68.2;
"бумага"	— все проводки, содержащие в содержании операции или в субконто или в реквизитах слово «бумага»;
"бумага"; "картон"	— все проводки, содержащие в содержании операции или в субконто или в реквизитах слово «бумага» или слово «картон»;
*, 46; "бумага"	— все проводки по кредиту 46 счета, содержащие в содержании операции или в субконто или в реквизитах слово «бумага»;

Кроме того, в обоих вариантах задаются дополнительные условия: валюта, план счетов, разделитель учета.

Параметр <Валюта> является значением типа «Справочник» вида, определенного при настройке валютного учета в метаданных. Если параметр указан, в отбор будут включены проводки только по указанной валюте. При этом пустое значение типа «Справочник» соответствующего вида считается указанием отбора по пустой валюте. Для того, чтобы отбор по валюте не производился нужно либо опустить данный параметр, либо передать туда значение иного типа, например, пустую сроку.

Параметр <ПланСчетов> является значением типа «План Счетов». Если параметр указан, в отбор будут включены проводки только по указанному плану счетов.

Параметр <РазделительУчета> является значением объекта метаданных, выбранного в качестве разделителя учета.

Если указан параметр <РазделительУчета>, в отбор будут включены проводки с указанным значением разделителя учета.

Параметры <Счет> и <КорСчет> задают условние отбора проводок по счетам.

Если указан параметр <Счет>, будут отбраны проводки только по указанному счету. Дополнительным условие отбора является значением параметра <Флаг>.

Если указаны параметр <Счет> и <КорСчет>, будут отобраны проводки только по указанной креспонденции счетов.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперацииСПроводками(ДатаНач, ДатаКон, "50, 51; ""По Чеку"" ");
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Сумма " + Опер.Сумма);
КонецЦикла;
```

ИспользоватьСубконто

Задает фильтр по субконто для функции ВыбратьОперацииСПроводками.

Синтаксис:

```
ИспользоватьСубконто(<ВидСубконто>, <Субконто>)
```

Англоязычный синоним:

```
UseSubconto
```

Параметры:

<ВидСубконто>	Значение типа «ВидСубконто» — отбор проводок будет выполнен только для субконто

	указанного вида.
<Субконто>	Значение субконто — отбор проводок будет выполнен только для указанного субконто. Кроме того, в качестве значения данного параметра можно передавать «Список значений».

Описание:

Метод ИспользоватьСубконто устанавливает фильтр по субконто, который используется при отборе проводок методом ВыбратьОперацииСПроводками.

Метод ИспользоватьСубконто может вызываться последовательно несколько раз. В этом случае фильтры, устанавливаемые этой функцией, суммируются.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ИспользоватьСубконто(ВидыСубконто.Организации, ВыбОрг);
Опер.ВыбратьОперацииСПроводками(ДатаНач, ДатаКон, "60");
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Сумма " + Опер.Сумма);
КонецЦикла;
```

ИспользоватьКорСубконто

Задает фильтр по корреспондирующим субконто для функции ВыбратьОперацииСПроводками.

Синтаксис:

```
ИспользоватьКорСубконто(<ВидСубконто>, <Субконто>)
```

Англоязычный синоним:

```
UseCorSubconto
```

Параметры:

<ВидСубконто>	Значение типа «ВидСубконто» — отбор проводок будет выполнен только для корреспондирующих субконто указанного вида.
<Субконто>	Значение субконто — отбор проводок будет выполнен только для указанного корреспондирующего субконто. Кроме того, в качестве значения данного параметра можно передавать «Список значений».

Описание:

Метод ИспользоватьКорСубконто устанавливает фильтр по корреспондирующим субконто, который используется при отборе проводок методом ВыбратьОперацииСПроводками.

Метод ИспользоватьКорСубконто может вызываться последовательно несколько раз. В этом случае фильтры, устанавливаемые этой функцией, суммируются.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ИспользоватьСубконто(ВидыСубконто.Организации, ВыбОрг);
Опер.ИспользоватьКорСубконто(ВидыСубконто.Товары, ВыбТовар);
Опер.ВыбратьОперацииСПроводками(ДатаНач, ДатаКон, "60");
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Сумма " + Опер.Сумма);
КонецЦикла;
```

ВыбратьПоЗначению

Открывает выборку операций или проводок, отобранных по значению отбора.

Синтаксис:

```
ВыбратьПоЗначению(<НачалоПериода>, <КонецПериода> , <ВидОтбора>, <Значение0тбора>)
```

Англоязычный синоним:

```
SelectByValue
```

Параметры:

<НачалоПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, с которого устанавливается начало выборки операций.
<КонецПериода>	Необязательный параметр. Выражение типа дата, документ или позиция документа, на котором устанавливается конец выборки операций.
<ВидОтбора>	Необязательный параметр. Символьная строка — название вида отбора (см. ниже).
<Значение0тбора>	Необязательный параметр. Значение отбора вида, указанного в параметре <ВидОтбора>.

Возвращаемое значение:

Число: 1 — действие выполнено и в выборке есть хотя бы одна операция или проводка; 0 — действие не выполнено или в выборке нет ни одной операции или проводки.

Описание:

Метод ВыбратьПоЗначению открывает выборку, содержащую операции или проводки за указанный период.

Данный метод позволяет достаточно быстро отобрать операции или проводки по критерию отбора. Возможные критерии отборов настраиваются в конфигураторе и имеют свои идентификаторы (системные или идентификаторы объектов метаданных). Вид отбора определяет будет открыта выборка операций или проводок.

Параметр <ВидОтбора> должен содержать название вида отборав виде символьной строки.

Параметр <ВидОтбора> для отбора операций может принимать следующие значения (в скобках указан английский синоним):

"СуммаОперации" ("OperSum")	Отбор по сумме операции. Доступно если в метаданных включен отбор по сумме операции.
"Содержание" ("Description")	Отбор по содержанию операции. Доступно если в метаданных включен отбор по содержанию операции.
<РеквизитОперации>	Отбор по дополнительному реквизиту операции. Идентификатор реквизита должен быть указан так, как это задано в метаданных. Доступно если в метаданных включен отбор по реквизиту операции.

Параметр <ВидОтбора> для отбора проводок может принимать следующие значения (в скобках указан английский синоним):

"Счет" ("Account")	Отбор по счету дебета или счету кредита проводки. Доступно если в метаданных включен отбор по счетам проводок.
"СчетДт" ("AccountDt")	Отбор по счету дебета проводки. Доступно если в метаданных включен отбор по дебету/кредиту счетов проводок.
"СчетКт" ("AccountKt")	Отбор по счету кредита проводки. Доступно если в метаданных включен отбор по дебету/кредиту счетов проводок.
"Валюта" ("Currency")	Отбор по валюте проводки. Доступно если в метаданных включен отбор по валюте проводок.
<ВидСубконто>	Отбор по виду субконто. Идентификатор вида субконто должен быть указан так, как он задан в метаданных. Доступно если в метаданных включен отбор по этому виду субконто.
<РеквизитПроводки>	Отбор по дополнительному реквизиту проводки. Идентификатор реквизита должен быть указан так, как он задан в метаданных. Доступно если в метаданных включен отбор по этому реквизиту проводки.

Параметр <ЗначениеОтбора> задает значение отбора.

После выполнения данного метода обход операций (проводок) осуществляется методами ПолучитьОперацию и ПолучитьПроводку.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьПоЗначению(ДатаНач, ДатаКон, "Счет", СчетПоКоду("60"));
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Сумма " + Опер.Сумма);
КонецЦикла;
```

ПолучитьОперацию

Извлекает операции из выборки, открытой при помощи метода ВыбратьОперции.

Синтаксис:

ПолучитьОперацию()

Англоязычный синоним:

GetOper

Возвращаемое значение:

Число: 1 — операция выбрана успешно; 0 — операция не выбрана (отсутствует).

Описание:

Метод ПолучитьОперацию выбирает очередную операцию из выборки, открытой при помощи метода ВыбратьОперации.

Метод ПолучитьОперацию используется для организации цикла по операциям. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьОперацию возвращает 1.

Метод возвращает 0, когда очередная операция не выбрана. Это происходит, если при предыдущем применении метода была выбрана последняя операция выборки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
```

```
   Сообщить("Операция " + Опер.Содержание);
КонецЦикла;
```

НайтиОперацию

Позиционирует объект по значению типа «Документ».

Синтаксис:

```
НайтиОперацию([<Документ>])
```

Англоязычный синоним:

```
FindOper
```

Параметры:

<Документ>	Значение типа «Документ».

Возвращаемое значение:

Число: 1 — действие выполнено, операция найдена; 0 — действие не выполнено, операция не найдена.

Описание:

Метод НайтиОперацию осуществляет поиск операции по значению типа «Документ».

В системе 1С:Предприятие каждая операция принадлежит документу. Причем операция принадлажит только одному документу, а у документа может существовать только одна операция. Операции введенные вручную принадлежат документам специального вида «Операция».

Поэтому не существует возможности передачи такого значения как «Операция». Для «идентификации» операции фактически используется значение документа, которому она принадлежит. Поэтому для позиционировании объекта «Операция» непосредственно на конкретную операцию используется значение типа «Документ».

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
```

Выбрана

Определяет выбрана ли в данный момент операция.

Синтаксис:

```
Выбрана()
```

Англоязычный синоним:

```
Selected
```

Возвращаемое значение:

Число: 1 — операция выбрана; 0 — операция не выбрана.

Описание:

При использовании объекта «Операция» созданного функцией СоздатьОбъект метод Выбрана определяет спозиционирован ли объект в настоящий момент на некоторой операции или нет.

При использовании данного метода к атрибуту документа Операция (который имеет тип «Операция») он определяет, существует ли реально операция у документа или нет.

При использовании в форме журнала операций (проводок) метод определяет спозиционирован ли курсор.

При использовании в форме операции метод определяет, записана новая операция или нет.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Если Опер.Выбрана() = 0 Тогда
   Сообщить("Операция не найдена");
КонецЕсли;
```

УстановитьАтрибут

Установить значение реквизита по имени идентификатора.

Синтаксис:

```
УстановитьАтрибут(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SebAttrib
```

Параметры:

<ИмяРеквизита>	Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.
<Значение>	Выражение, содержащее устанавливаемое значение реквизита.

Описание:

Метод УстановитьАтрибут позволяет установить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
Опер.УстановитьАтрибут("ЦенаРозн", ЦенаТов);
```

ПолучитьАтрибут

Получить значение реквизита по имени идентификатора.

Синтаксис:

```
ПолучитьАтрибут(<ИмяРеквизита>)
```

Англоязычный синоним:

```
GetAttrib
```

Параметры:

<ИмяРеквизита> Строковое выражение, содержащее имя реквизита, как оно задано в конфигураторе.

Возвращаемое значение:

Значение реквизита <ИмяРеквизита>.

Описание:

Метод ПолучитьАтрибут позволяет получить значение реквизита по имени идентификатора, как оно задано в конфигураторе.

Пример:

```
ЦенаТов = Опер.ПолучитьАтрибут("ЦенаРозн");
```

ВыбратьПроводки

Открывает выборку проводок текущей операции.

Синтаксис:

```
ВыбратьПроводки()
```

Англоязычный синоним:

```
Select Entries
```

Возвращаемое значение:

Число: 1 — действие выполнено и в выборке есть хотя бы одна проводка;
0 —действие не выполнено или в выборке нет ни одной проводки.

Описание:

Метод ВыбратьПроводки открывает выборку проводок текущей операции. Данный метод предназначен для организации перебора проводок операции и используется вмесет с методом ПолучитьПроводку.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.Дебет.Счет + " – " + Опер.Кредит.Счет);
КонецЦикла;
```

ПолучитьПроводку

Получить проводку из выборки, открытой методом ВыбратьПроводки.

Синтаксис:

```
ПолучитьПроводку()
```

Англоязычный синоним:

```
GetEntry
```

Возвращаемое значение:

Число: 1 — следующая проводка выбрана успешно; 0 — следующая проводка не выбрана (отсутствует).

Описание:

Метод ПолучитьПроводку выбирает очередную проводку из выборки, содержащей проводки текущей операции. Перед применением метода ПолучитьПроводку выборка должна быть открыта при помощи метода ВыбратьПроводки.

Метод ПолучитьПроводку используется для организации цикла по проводкам текущей операции. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьПроводку возвращает 1.

Метод возвращает 0, когда очередная проводка не выбрана. Это происходит, если при предыдущем применении метода была выбрана последняя проводка выборки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
```

```
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.Дебет.Счет + " - " + Опер.Кредит.Счет);
КонецЦикла;
```

ПроводкаВыбрана

Определяет выбрана ли проводка операции.

Синтаксис:

```
ПроводкаВыбрана()
```

Англоязычный синоним:

```
EntrySelected
```

Возвращаемое значение:

Число: 1 — проводка выбрана; 0 — проводка не выбрана.

Описание:

Метод `ПроводкаВыбрана` позволяет определить, спозиционирована ли в настоящий момент некоторая проводка в операции или нет, то есть можно ли обращаться к атрибутам и методам проводки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Опер.ПолучитьПроводку();
Если Опер.ПроводкаВыбрана() = 0 Тогда
   Сообщить("Нет проводок в операции");
КонецЕсли;
```

КоличествоПроводок

Оперделяет количество проводок в операции включая и проводки и корреспонденции.

Синтаксис:

```
КоличествоПроводок()
```

Англоязычный синоним:

```
EntriesCount
```

Возвращаемое значение:

Целое число — количество проводок в текущей операции.

Описание:

Метод позволяет опередделить полное количество проводок и корреспонденции в операции.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
КонецЦикла;
```

ПолучитьПроводкуПоНомеру

Выбрать в качестве текущей проводку по ее номеру и номеру корреспонденции.

Синтаксис 1:

```
ПолучитьПроводкуПоНомеру(<АбсолютныйНомерПроводки>)
```

Синтаксис 2:

```
ПолучитьПроводкуПоНомеру(<НомерПроводки>, <НомерКорреспонденции>)
```

Англоязычный синоним:

```
GetEntryByNumber
```

Параметры:

`<АбсолютныйНомерПроводки>`	Необязатальный параметр. Число — абсолютный номер проводки в операции (включая проводки и корреспонденции). Если не задан –1.
`<НомерПроводки>`	Необязатальный параметр. Число — номер проводки в операции.
`<НомерКорреспонденции>`	Необязатальный параметр. Число — номер корреспонденции в проводке. Параметр имеет смысл, если указан параметр `<НомерПроводки>`.

Возвращаемое значение:

Число: 0 — проводка не найдена; 1 — проводка найдена.

Описание:

Метод `ПолучитьПроводкуПоНомеру` выбирает в качестве текущей проводку или корреспонденцию проводки.

Если передан один параметр, то он воспринимается, как абсолютный номер проводки (корреспонденции) в операции среди всех проводок и корреспонденции. Если передано два параметра, то первый из них — это номер проводки, а второй — номер корреспонденции в проводке.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ПолучитьПроводкуПоНомеру(3, 2);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
КонецЦикла;
```

Пров

Обращение к данным проводки по номеру.

Синтаксис:

```
Пров(<НомерПроводки>, <НомерКорреспонденции>)
```

Англоязычный синоним:

```
Entry
```

Параметры:

<НомерПроводки>	Необязательный параметр. Число — номер проводки в операции. Если параметр не задан, то используется текущая проводка.
<НомерКорреспонденции>	Необязательный параметр. Число — номер корреспонденции в проводке. Параметр имеет смысл, если указан параметр <НомерКорреспонденции>.

Возвращаемое значение:

Ссылка на указанную проводку.

Описание:

Метод Пров является специальным методом, позволяющим обратиться непосредственно к данным проводки операции по номеру, без установки текущей проводки. Возвращаемое методом значение является ссылкой на проводку. Оно не может использоваться как значение, а предназначено только для доступа к атрибутам проводки. В основном метод Пров имеет смысл применять в формулах типовых операций.

Метод Пров позволяет обратится к атрибутам проводки Сумма, СуммаВал, Количество, Валюта, Дебет, Кредит, <Реквизит>.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Сообщить(Пров(2, 3).Дебет.Счет);
```

НомерПроводки

Определяет номер текущей проводки.

Синтаксис:

```
НомерПроводки()
```

Англоязычный синоним:

```
EntryNumber
```

Возвращаемое значение:

Число — номер проводки.

Описание:

Для текущей проводки (корреспонденции) возвращает ее номер (не абсолютный, а именно номер проводки без учета корреспонденции).

Пример:

```
Опер = СоздатьОбъект ("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
   Сообщить("Проводка " + Опер.НомерПроводки() +
            "Корр. " + Опер.НомерКорреспонденции());
КонецЦикла;
```

ПланСчетов

Выдает план счетов текущей проводки.

Синтаксис:

```
ПланСчетов()
```

Англоязычный синоним:

ChartOfAccounts
Возвращаемое значение:
Значение типа «План Счетов».
Описание:
Для текущей проводки возвращает ее план счетов.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
   Сообщить("Проводка " + Опер.НомерПроводки() + "Корр. " +
            Опер.НомерКорреспонденции() + "Пл.сч." + Опер.ПланСчетов());
КонецЦикла;
```

НомерКорреспонденции

Определяет номер текущей корреспонденции в операции.
Синтаксис:
НомерКорреспонденции()
Англоязычный синоним:
CorrespondenceNumber
Возвращаемое значение:
Число — номер текущей корреспонденции в проводке.
Описание:
Для текущей проводки (корреспонденции) возвращает ее номер корреспонденции, если проводка не сложная — 1.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
   Сообщить("Проводка " + Опер.НомерПроводки() +
            "Корр. " + Опер.НомерКорреспонденции());
КонецЦикла;
```

СложнаяПроводка

Определяет является ли текущая проводка сложной.
Синтаксис:
СложнаяПроводка()
Англоязычный синоним:
ComplexEntry
Возвращаемое значение:
Число: 1 — текущая проводка является сложной; 0 — текущая проводка не является сложной.
Описание:
Для текущей проводки (корреспонденции) определяется является ли она сложной. Под сложной проводкой понимается проводка, состоящая из нескольких корреспонденции.
Пример:
```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Для Инд = 1 До Опер.КоличествоПроводок() Цикл
   Опер.ПолучитьПроводкуПоНомеру(Инд);
   Если Опер.СложнаяПроводка() = 1 Тогда
      Сообщить("Проводка сложная");
   КонецЕсли;
КонецЦикла;
```

НомерСтрокиДокумента

Выдает номер строки документа, по которой сформирована текущая проводка.
Синтаксис:
НомерСтрокиДокумента()
Англоязычный синоним:
DocLineNum
Возвращаемое значение:

Число — номер строки документа, сформировавшей данную проводку.

Описание:

При добавлении проводок в операцию в процессе проведения документа существует возможность привязать каждую проводку к определенной строке документа. Это позволяет в дальнейшем получить из документа по номеру строки дополнительную информацию о проводке. Для того, что в проводке был простален номер строки документа следует в модуле документа перед добавлением проводки в операцию использовать метод ПривязыватьСтроку.

Метод НомерСтрокиДокумента выдает для текущей проводки номер строки документа, который был привязан в момент добавления проводки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Номер строки " + Опер.НомерСтрокиДокумента());
КонецЦикла;
```

ПредставлениеПроводки

Получить представление для текущей проводки.

Синтаксис:

```
ПредставлениеПроводки(<ПоСубконто>)
```

Англоязычный синоним:

```
Entry Presentation
```

Параметры:

<ПоСубконто> Необязательный параметр. Флаг включения в представление проводки представления субконто проводки. Число:

0 — не включать представление субконто;

1 — включать представление субконто;

2 — включать развернутое представление субконто. Значение по умолчанию: 0.

Возвращаемое значение:

Строка — представление для текущей проводки.

Описание:

Представлением называется символьная строка, содержащая информацию из реквизитов проводки и операции. Эта строка может быть использована для отображения проводки в различных отчетах, диалогах и других визуальных элементах конфигурации. Формат представления определяется в конфигураторе при редактировании свойств проводки.

Метод ПредставлениеПроводки позволяет получить представление для текущей проводки.

Параметр метода позволяет указать, включать ли в представление проводки и представление субконто проводки. Представление субконо содержит информацию из реквизитов значения субконто. Представление может быть задано в конфигураторе при редактировании свойств вида субконто.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Проводка " + Опер.ПредставлениеПроводки(1));
КонецЦикла;
```

ПредставлениеСубконто

Получить представление для субконто текущей проводки.

Синтаксис:

```
ПредставлениеСубконто()
```

Англоязычный синоним:

```
SubcontoPresentation
```

Возвращаемое значение:

Символьная строка — представление субконто текущей проводки.

Описание:

Представлением называется символьная строка, содержащая информацию из реквизитов субконто. Эта символьная строка может быть использована для отображения значений реквизитов субконто в различных отчетах, диалогах и других визуальных элементах конфигурации.

Представление может быть задано в конфигураторе при редактировании свойств вида субконто.

Метод ПредставлениеСубконто позволяет получить представление для всех субконто текущей проводки.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДокумент);
Опер.ВыбратьПроводки();
Пока Опер.ПолучитьПроводку() = 1 Цикл
   Сообщить("Субконто " + Опер.ПредставлениеСубконто());
КонецЦикла;
```

НазначитьТип

Назначить тип для реквизита неопределенного вида.

Синтаксис:

```
НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
SetType
```

Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита операции или проводки неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту операции или проводки. Например: "Строка", "Число", "Справочник.Товары", "Документ.РасходнаяНакладная" и т. п.
<Длина>	Необязательный параметр. Числовое выражение — длина поля представления данных. Имеет смысл только при задании числового или строкового типа.
<Точность>	Необязательный параметр. Числовое выражение — число знаков числа после десятичной точки. Имеет смысл только при задании числового типа.

Описание:

Метод НазначитьТип позволяет назначить тип для реквизита, которому в конфигураторе назначен тип «Неопределенный».

Пример:

```
Операция.НазначитьТип("ТМЦ", "Справочник.Товары");
```

Новая

Создать новую операцию.

Синтаксис:

```
Новая()
```

Англоязычный синоним:

```
New
```

Описание:

Метод Новая используется для создания операций из встроенного языка, не принадлежащих документу какого либо вида.

Такая операция будет являться полным аналогом операции введенной вручную, то есть будет принадлежать документу специального вида «Операция». Операции, принадлежащие документам различных видов записываются при проведении документа без использования метода Новая.

Метод Новая может быть вызван только для объектов типа «Операция» созданных функцией СоздатьОбъект.

После вызова метода Новая могут быть заданы значения реквизитов операции и добавлены проводки. После этого для записи операции должен быть вызван метод Записать.

Пример:

* Создание и запись новой операции созданной в модуле отчета или обработки

```
Опер = СоздатьОбъект("Операция");
Опер.Новая();
Опер.Дата = РабочаяДата();
Опер.Содержание = "Выдача денег из кассы";
Опер.Записать();
```

ЗаписатьПроводки

Записать в информационную базу уже добавленные и заполненные проводки операции.

Синтаксис:

```
ЗаписатьПроводки()
```

Англоязычный синоним:

```
WriteEntries
```

Описание:

Метод ЗаписатьПроводки выполняет запись в информационную базу уже добавленных и заполненных проводок операции. Метод может использоваться только для атрибута «Операция» документа в момент проведения (в процедуре ОбработкаПроведения). При этом происходит обновление бухгалтерских итогов. Это позволяет при проведении документа обращаться к бухгалтерским итогам уже измененным проводками, записанными этим документом. После выполнения метода ЗаписатьПроводки и до окончания процедуры ОбработкаПроведение уже невозможно изменять или удалять проводки, добавленные до вызова этого метода.

Данный метод имеет смысл использовать, только в том случае, если существует необходимость обращения к итогам, измененным проводками записываемой операции.

Замечание. При записи сложной проводки, если у главной корреспонденции сложной проводки не указана сумма (равна 0), то она автоматически вычисляется на основании подчиненных корреспонденции.

Пример:

```
Процедура СписаниеСчетаНаСч20(Счт)
   // создаем и формируем проводки
КонецПроцедуры

Процедура СписаниеСчета20На40()
   // создаем и формируем проводки
КонецПроцедуры

Процедура ОбработкаПроведения()
   Сч20 = СчетПоКоду("20");
   Сч40 = СчетПоКоду("40");
   СписаниеСчетаНаСч20(СчетПоКоду("25"));
   СписаниеСчетаНаСч20(СчетПоКоду("26"));
   Операция.ЗаписатьПроводки();
   СписаниеСчета20На40();
   Операция.Содержание = "Закрытие фин. результатов за " +
                         Формат(ДатаДок, "Д ММММГГГГ");
   Операция.Записать();
КонецПроцедуры
```

Записать

Записать измененную или новую операцию.

Синтаксис:

```
Записать()
```

Англоязычный синоним:

```
Write
```

Описание:

Метод Записать выполняет запись новой или измененной операции.

Он может быть использован в следующих ситуациях:

- метод Записать может быть вызван для записи новой или измененной ручной операции (принадлежащей документу специального вида «Операция»). Такие операции могут создаваться и редактироваться пользователем вручную или из встроенного языка объектом «Операция», созданным функцией СоздатьОбъект.
- метод Записать может быть вызван для записи операции созданной при проведении документа в предопределенной процедуре ОбработкаПроведения. Данный метод должен быть вызван после заполнения операции содержанием реквизитов и проводками.
- этот метод может применяться в Модуле формы операции непосредственно к операции локального контекста, в этом случае данный метод отрабатывает те же действия, как интерактивное нажатие пользователем кнопки с формулой "#Записать".

Замечание. При записи сложной проводки, если у главной корреспонденции сложной проводки не указана сумма (равна 0), то она автоматически вычисляется на основании подчиненных корреспонденции.

Пример:

* Создание и запись новой операции созданной в модуле отчета или обработки.

```
Опер = СоздатьОбъект("Операция");
Опер.Новая();
Опер.Дата = РабочаяДата();
Опер.Содержание = "Деньги по чеку";
```

```
Опер.НоваяПроводка();
Опер.Дебет.Счет = СчетПоКоду("50");
Опер.Кредит.Счет = СчетПоКоду("51");
Опер.Сумма = 1000000;
Опер.Записать();
```

* Запись операции в модуле документа в процессе проведения

```
Процедура ОбработкаПроведения()
   Операция.Содержание = "Выдача денег из кассы";
   Операция.НоваяПроводка();
   Операция.Дебет.Счет = СчетПоКоду("71");
   Операция.Дебет.Сотрудники = Сотрудник;
   Операция.Кредит.Счет = СчетПоКоду("50");
   Операция.Сумма = СуммаВыдачи;
   Операция.Записать();
КонецПроцедуры
```

Удалить

Удаление операции.

Синтаксис:

`Удалить(<Режим>)`

Англоязычный синоним:

`Delete`

Параметры:

`<Режим>` Необязательный параметр. Числовое выражение:
1 — непосредственное удаление; 0 — пометка на удаление. Значение по умолчанию — 1.

Описание:

Метод `Удалить` удаляет (или помечает на удаление) текущую операцию.

Замечание. Непосредственное удаление объекта следует применять только в особых случаях, так как могут существовать ссылки на удаляемое значение в документах, справочниках и т. д. и непосредственное удаление может повлечет нарушение ссылочной целостности.

Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`.

Если метод `Удалить` применен к операции принадлежащей документу не специального вида «Операция», то происходит удаление (или пометка) всего документа.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДок);
Опер.Удалить(0);
```

ПометкаУдаления

Проверяет наличие пометки на удаление текущей операции.

Синтаксис:

`ПометкаУдаления()`

Англоязычный синоним:

`DeleteMark`

Возвращаемое значение:

Число: 1 — операция помечена на удаление; 0 — операция не помечена на удаление.

Описание:

Метод используется для проверки, помечена ли на удаление текущая операция или документ, которому принадлежит операция.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Если Опер.ПометкаУдаления() = 1 Тогда
      Сообщить(Опер.Документ);
   КонецЕсли;
КонецЦикла;
```

СнятьПометкуУдаления

Снять пометку удаления операции.

Синтаксис:

```
СнятьПометкуУдаления()
```

Англоязычный синоним:

```
ClearDeleteMark
```

Описание:

Метод `СнятьПометкуУдаления` снимает пометку удаления текущей операции.

Данный метод может использоваться только для объектов, созданных функцией `СоздатьОбъект`. Если метод `СнятьПометкуУдаления` применен к операции принадлежащей документу не специального вида «Операция», то происходит снятие пометки удаления всего документа.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Если Опер.ПометкаУдаления() = 1 Тогда
      Опер.СнятьПометкуУдаления();
   КонецЕсли;
КонецЦикла;
```

ПолучитьВремя

Возвращает время текущей операции.

Синтаксис:

```
ПолучитьВремя(<Часы>, <Минуты>, <Секунды>)
```

Англоязычный синоним:

```
GetTime
```

Параметры:

`<Часы>`	Необязательный параметр. Идентификатор переменной, в которую метод возвращает число — час времени операции.
`<Минуты>`	Необязательный параметр. Идентификатор переменной, в которую метод возвращает число — минуты времени операции.
`<Секунды>`	Необязательный параметр. Идентификатор переменной, в которую метод возвращает число — секунды времени операции.

Возвращаемое значение:

Строковое значение времени в виде "ЧЧ.ММ.СС".

Описание:

Метод `ПолучитьВремя` возвращает время операции. Время записывается в переменные, передаваемые как параметры при вызове метода.

Пример:

```
Перем Час;
Перем Минута;
Перем Секунда;
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Опер.ПолучитьВремя(Час, Минута, Секунда);
КонецЦикла;
```

УстановитьВремя

Задает время текущей операции.

Синтаксис:

```
УстановитьВремя(<Часы>, <Минуты>, <Секунды>)
```

Англоязычный синоним:

```
SetTime
```

Пареметры:

`<Часы>`	Необязательный параметр. Число — час операции.
`<Минуты>`	Необязательный параметр. Число — минуты операции.
`<Секунды>`	Необязательный параметр. Число — секунды операции.

Описание:

Метод `УстановитьВремя` изменяет время операции.

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект. Так как время является значением относящимся к документу, то вызов этого метода изменяет время собственно документа, которому принадлежит операция.

Пример:

```
Перем Час;
Перем Минута;
Перем Секунда;
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Опер.ПолучитьВремя(Час, Минута, Секунда);
   Опер.УстановитьВремя(Час + 1, Минута, Секунда);
КонецЦикла;
```

ПолучитьДокумент

Возвращает значение типа «Документ» содержащий документ, которому принадлежит операция.

Синтаксис:

```
ПолучитьДокумент()
```

Англоязычный синоним:

```
GetDocument
```

Возвращаемое значение:

Значение типа «Документ».

Описание:

Метод ПолучитьДокумент возвращает значение типа «Документ» содер-жащий документ, которому принадлежит операция. Его следует использовать, если нужно передать какому либо методу или запомнить в переменную значение типа документ операции. Для доступа к данным документа непосредственно из объекта типа «Операция» следует использовать атрибут объекта «Документ».

Пример:

```
Опер = СоздатьОбъект("Операция");
Док = СоздатьОбъект("Документ");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Сообщить("Номер " = Опер.Документ.Номер);
   Док.НайтиДокумент(Опер.ПолучитьДокумент());
КонецЦикла;
```

ВключитьПроводки

Включает/выключает проводки операции.

Синтаксис:

```
ВключитьПроводки(<Флаг>)
```

Англоязычный синоним:

```
EntriesOn
```

Пареметры:

<Флаг>	Необязательный параметр. 1 — включить проводки операции. 0 — выключить проводки операции. Если параметр не передан — состояние не изменяется.

Возвращаемое значение:

Состояние на момент вызова метода. 1 — проводки операции включены; 0 — проводки операции выключениы.

Описание:

Для каждой бухгалтерской операции может быть выполнено выключение проводок. Это значит, что проводки остаются, но исключаются из итогов. Эти действия могут выполняться пользователем интерактивно и методом ВключитьПроводки для объекта типа «Операция».

Данный метод может использоваться только для объектов, созданных функцией СоздатьОбъект. Изменяется состояние всех выбранной проводок операции одновременно.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.ВыбратьОперации(ДатаНач, ДатаКон);
Пока Опер.ПолучитьОперацию() = 1 Цикл
   Опер.ВключитьПроводки(0);
КонецЦикла;
```

НоваяПроводка

Создать новую проводку.

Синтаксис:

НоваяПроводка()

Англоязычный синоним:

NewEntry

Описание:

Метод создает новую проводку для текущей операции. Новая проводка становится текущей.

Во-первых, метод может быть вызван для объекта «Операция», созданного функцией СоздатьОбъект, при создании или изменении операции принадлежащей документу специального вида «Операция».

Во-вторых, метод может быть вызван для объекта, являющегося атрибутом документа, при создании операции принадлежащей документу, в процессе проведения документа.

Созданная проводка будет записана и повлечет изменения в бухгалтерских итогах после вызова метода операции Записать.

Пример:

* Создание и запись новой операции созданной в модуле отчета или обработки

```
Опер = СоздатьОбъект("Операция");
Опер.Новая();
Опер.Дата = РабочаяДата();
Опер.Содержание = "Деньги по чеку";
Опер.НоваяПроводка();
Опер.Дебет.Счет = СчетПоКоду("5 0");
Опер.Кредит.Счет = СчетПоКоду("51");
Опер.Сумма = 1000000;
Опер.Записать();
```

* Запись операции в модуле документа в процессе проведения

```
Процедура ОбработкаПроведения()
   Операция.Содержание = "Выдача денег из кассы";
   Операция.НоваяПроводка();
   Операция.Дебет.Счет = СчетПоКоду("71");
   Операция.Дебет.Сотрудники = Сотрудник;
   Операция.Кредит.Счет = СчетПоКоду("50");
   Операция.Сумма = СуммаВыдачи;
   Операция.Записать();
КонецПроцедуры
```

НоваяКорреспонденция

Создает новую корреспонденцию проводки.

Синтаксис:

НоваяКорреспонденция()

Англоязычный синоним:

NewCorrespondence

Описание:

Метод НоваяКорреспонденция создает новую корреспонденцию для проводки. Метод должен использоваться после того, как новая проводка создана с использованием метода НоваяПроводка.

Данный метод предназначен для формирования сложных проводок, состоящих из нескольких корреспонденции.

Для проводки может быть создано практически неограниченное число корреспонденции путем последовательного вызова метода НоваяКорреспонденция. После создания корреспонденции изменение атрибутов проводки вызывает изменение атрибутов новой корреспонденции.

Пример:

```
Процедура ОбработкаПроведения()
   Операция.Содержание = "Выдача денег из кассы";
   Операция.НоваяПроводка();
   Операция.Кредит.Счет = СчетПоКоду("50");
   Операция.Сумма = Итог("СуммаВыдачи");
   ВыбратьСтроки();
   Пока ПолучитьСтроку() = 1 Цикл
      Операция.НоваяКорреспонденция();
      Операция.Дебет.Счет = СчетПоКоду("71");
      Операция.Дебет.Сотрудники = Сотрудник;
      Операция.Сумма = СуммаВыдачи;
```

```
   КонецЦикла;
   Операция.Записать();
КонецПроцедуры
```

ПроверитьПроводку

Проверить проводку на соответствие корректным проводкам.

Синтаксис:

```
ПроверитьПроводку()
```

Англоязычный синоним:

```
CheckEntry
```

Возвращаемое значение:

Число: 1 — проводка соответствует корректным проводкам; 0 — не соответствует.

Описание:

Метод `ПроверитьПроводку` проверяет проводку на соответствие корректным проводкам.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДок);
Пока Опер.КоличествоПроводок() > 0 Цикл
   Опер.ПолучитьПроводкуПоНомеру(1);
   Если Опер.ПроверитьПроводку() = 0 Тогда
      Опер.УдалитьПроводку();
   КонецЕсли;
КонецЦикла;
Опер.Записать();
```

УдалитьПроводку

Удалить проводку.

Синтаксис:

```
УдалитьПроводку()
```

Англоязычный синоним:

```
DeleteEntry
```

Описание:

Метод удаляет текущую проводку или корреспонденцию операции. Метод может быть вызван объектом «Операция», созданным функцией `СоздатьОбъект`, при создании или изменении операции принадлежащей документу специального вида «Операция».

Собственно изменение в составе проводок будут записаны и повлекут из-менния в бухгалтерских итогах после вызова метода операции `Записать`.

Пример:

```
Опер = СоздатьОбъект("Операция");
Опер.НайтиОперацию(ВыбрДок);
Пока Опер.КоличествоПроводок() > 0 Цикл
   Опер.ПолучитьПроводкуПоНомеру(1);
   Опер.УдалитьПроводку();
КонецЦикла;
Опер.Записать();
```

Атрибуты контекста модуля формы операции

БИ

Значение, содержащее агрегатный объект типа «БухгалтерскиеИтоги»

Синтаксис:

```
БИ
```

Англоязычный синоним:

```
AT
```

Описание:

Данный объект включается в контекст формы операции для облегчения доступа в итогам при работе с типовыми операциями. Это позволяет обращаться в формулах типовых операций к этому объекту для получения итогов.

Пример:

* Формула суммы проводки типовой опеарции

```
БИ.ПериодМ(ДатаОперации);
```

```
БИ.СКД(Дебет.Счет)
```

Методы контекста модуля формы операции

ПоТиповойОперации

Определяет, вводится ли новая операция с использованием типовой, и какая типовая операция используется.

Синтаксис:

```
ПоТиповойОперации (<Переменная>)
```

Англоязычный синоним:

```
ByTemplateOper
```

Параметры:

<Переменная>	Необязательный параметр. Имя переменной, в которую будет записано наименование типовой операции.

Возвращаемое значение:

Число: 1 — если при вводе операции вручную использована типовая опера-ци; 0 — если при вводе операции вручную не использовалась типовая операция.

Описание:

Определяет, вводится ли новая операция с использованием типовой. Если указан параметр <Переменная>, в переменную будет записано имя типовой операции;

Пример:

```
Процедура ПриОткрытии()
   Если ПоТиповойОперации() = 0 Тогда
      ИспользоватьВалюту(Константа.ОснВалюта);
   КонецЕсли;
КонецПроцедуры;
```

ИспользоватьВалюту

Установка валюты используемой по умолчанию.

Синтаксис:

```
МспользоватьВалюту(<Валюта>)
```

Англоязычный синоним:

```
UseCurrency
```

Параметры:

<Валюта>	Необязательный параметр. Значение типа «Справочник», имеющий вид, определенный при настройке валютного учета. Если параметр не задан, установка не изменяется.

Возвращаемое значение:

Значение установки используемой валюты на момент до вызова метода.

Описание:

Данный метод устанавливает в форме используемую по умолчанию валюту. Она будет автоматически подставляться в валюту вводимых проводок, если в проводках используются валютные счета. После автоматической подстановки пользователь может изменить валюту по своему усмотрению.

Пример:

```
Процедура ПриОткрытии()
   ИспользоватьВалюту(Константа.ОснВалюта);
КонецПроцедуры;
```

ИспользоватьКорСчет

Установка корреспондирующего счета используемого по умолчанию.

Синтаксис:

```
ИспользоватьКорСчет(<Счет>)
```

Англоязычный синоним:

```
UseCorAccount
```

Параметры:

<Счет>	Необязательный параметр. Значение типа «Счет». Если параметр не задан, установка не изменяется.

Возвращаемое значение:

Значение установки используемого по умолчанию корреспондирующего счета на момент до вызова метода.

Описание:

Данный метод устанавливает в форме используемый по умолчанию корреспондирующий счет. Он будет автоматически подставляться проводку взамен не указанных пользователем счетов. Эта возможность может быть использована, например, для ввода первоначальных остатков для автоматической простановки счета, корреспондирующего с введенным.

Пример:

```
Процедура ПриОткрытии()
   ИспользоватьКорСчет(Константа.СчетПервОстатков);
КонецПроцедуры;
```

ИспользоватьСубконто

Установка субконто используемого по умолчанию.

Синтаксис:

```
ИспользоватьСубконто(<ВидСубконто>, <Субконто>)
```

Англоязычный синоним:

```
UseSubconto
```

Параметры:

<ВидСубконто>	Значение типа «ВидСубконто».
<Субконто>	Необязательный параметр. Значение субконто. Если параметр не задан, установка не изменяется.

Возвращаемое значение:

Значение установки используемого по умолчанию субконто на момент до вызова метода.

Описание:

Данный метод устанавливает в форме используемые по умолчанию значения субконто разных видов. Они будут автоматически подставляться субконто проводок, если в проводках используются счета с этими видами субконто. После автоматической подстановки пользователь может изменить субконто по своему усмотрению.

Пример:

```
Процедура ПриОткрытии()
   ИспользоватьСубконто(ВидыСубконто.Склады, Константа.ОснСклад);
КонецПроцедуры;
```

ИзменениеПорядкаСтрок

Установить режим изменения порядка строк в форме операции.

Синтаксис:

```
ИзменениеПорядкаСтрок(<Разрешить>)
```

Англоязычный синоним:

```
ChangeLinesOrder
```

Параметры:

<Разрешить>	Необязательный параметр. Число: 1 — разрешить изменение порядка строк в операции; 0 — запретить. Если параметр не задан, то режим не меняется.

Возвращаемое значение:

Режим изменения порядка строк до исполнения метода. Число: 1 — разрешено изменение порядка строк в операции; 0 — запрещено.

Описание:

Метод `ИзменениеПорядкаСтрок` позволяет установить режим изменения порядка строк в форме операции.

Пример:

```
ИзменениеПорядкаСтрок(0);
```

Предопределенные процедуры модуля формы операции

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в модуле формы операции (см. «Виды программных модулей»).

В основном данные процедуры предназначены для расширения возможности программного управления правами доступа к системе.

Предопределенные процедуры не является встроенными процедурами языка. Для них зарезервированы только название и синтаксис. Тело процедур должно быть написано самим разработчиком в соответствующих программных модулях. Вызов предопределенных процедур на исполнение производится в системе 1С:Предприятие неявно при возникновении соответствующего события. Описание предопределенных процедур также см. гл. «Системные предопределенные процедуры».

ВводНового

Предопределенная процедура при вводе новой операции.

Синтаксис:

```
ВводНового(<ПризнКопирования>, <ОбъектКопирования>)
```

Англоязычный синоним:

```
InputNew
```

Параметры:

<ПризнКопирования>	Признак того, что объект введен копированием. Число: 1 — объект введен копированием, 0 — просто новый объект. Данный признак может быть использован для анализа необходимости инициализации реквизитов нового объекта.
<ОбъектКопирования>	Объект, который был скопирован.

Описание:

Вызов процедуры ВводНового производится в системе 1С:Предприятие неявно в момент выбора пункта «Новая» из меню «Действия» главного меню системы 1С:Предприятие при работе с журналом операций, или в других случаях при открытии формы операции для ввода новой операции. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов новой операции. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя вводить операции), ввода новой операции и открытие ее формы не будет выполнено.

Процедура ВводНового должна размещаться в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНового()
   ИспользоватьВалюту(Константа.ОснВалюта);
КонецПроцедуры
```

См. также: СтатусВозврата

ВводНаОсновании

Предопределенная процедура при вводе новой операции на основании.

Синтаксис:

```
ВводНаОсновании(<ДокОснование>)
```

Англоязычный синоним:

```
InputCausedBy
```

Параметры:

<ДокОснование>	Значение документа, на основании которого вводится новая операция.

Описание:

Вызов процедуры ВводНаОсновании производится в системе 1С:Предприятие неявно после выбора пункта «Ввести на основании» из меню «Действия» главного меню системы 1С:Предприятие при работе с журналом операций или в других случаях, когда форма открывается для ввода на основании. В этот момент система подставляет фактическое значение параметра <ДокОснование>, содержащее объект типа «Документ», на которой находился курсор в момент выполнения данной команды.

Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) реквизитов новой операции, вводимых на основании выбранного документа

Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя вводить новые операции), ввода новой операции и открытие формы не будет выполнено.

Процедуру ВводНаОсновании можно размещать только в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ВводНаОсновании(ДокОсн)
   Если ДокОсн.Вид() = "ПлатПор" Тогда
      ИспользоватьСубконто(ВидыСубконто.Организации, ДокОсн.Получатель);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗаписи

Предопределенная процедура вызывается при записи операции.

Синтаксис:

```
ПриЗаписи()
```

Англоязычный синоним:

```
OnWrite
```

Описание:

Вызов предопределенной процедуры ПриЗаписи производится в системе 1С:Предприятие при записи операции в форме операции. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если не правильно заполнены реквизиты операции), запись операции не будет выполнена.

Данная предопределенная процедура может располагаться в модулях формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриЗаписи()
   Если ПустаяСтрока(Содержание) = 1 Тогда
      Предупреждение("Не заполнено содержание операции!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриНачалеРедактированияСтроки

Предопределенная процедура вызывается при редактировании существующей проводки или корреспонденции операции.

Синтаксис:

```
ПриНачалеРедактированияСтроки()
```

Англоязычный синоним:

```
OnStartEditLine
```

Описание:

Вызов предопределенной процедуры ПриНачалеРедактированияСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования существующей проводки или корреспонденции операции.

Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя редактировать проводки операции), то проводка не будет изменена.

Данная предопределенная процедура может располагаться в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права изменять проводки!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВводеСтроки

Предопределенная процедура при вводе новой проводки или корреспонденции операции.

Синтаксис:

```
ПриВводеСтроки()
```

Англоязычный синоним:

```
OnNewLine
```

Описание:

Вызов предопределенной процедуры ПриВводеСтроки производится в системе ЮПредприятие при интерактивном вводе новой проводки или корреспонденции операции. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю редактировать операции), то новая строка не будет инициирована.

Данная предопределенная процедура может располагаться в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права добавлять проводки!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриРедактированииНовойСтроки

Предопределенная процедура при редактировании новой проводки или корреспонденции операции.

Синтаксис:

```
ПриРедактированииНовойСтроки()
```

Англоязычный синоним:

```
OnEditNewLine
```

Описание:

Вызов предопределенной процедуры ПриРедактированииНовойСтроки производится в системе 1С:Предприятие в момент начала интерактивного редактирования новой проводки или корреспонденции операции. Данная процедура может использоваться, например, для установки начальных значений (по умолчанию) проводки. В данной предопределенной процедуре установка статуса возврата не имеет смысла, т. к. отказаться от ввода новой проводки в этот момент уже невозможно.

Данная предопределенная процедура может располагаться в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриРедактированияНовойСтроки()
   Фирма = Константа.ОснФирма;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУдаленииСтроки

Предопределенная процедура при удалении проводки или корреспонденции операции.

Синтаксис:

```
ПриУдаленииСтроки()
```

Англоязычный синоним:

```
OnDeleteLine
```

Описание:

Вызов предопределенной процедуры ПриУдаленииСтроки производится в системе 1С:Предприятие при интерактивном удалении проводки или корреспонденции операции.

Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя редактировать проводки операции), удаление проводки или корреспонденции операции не будет выполнено.

Данная предопределенная процедура может располагаться только в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриНачалеРедактированияСтроки()
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Предупреждение("У вас нет права изменять проводки!");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриИзмененииПорядкаСтрок

Предопределенная процедура при изменении порядка строк операции.

Синтаксис:

```
ПриИзмененииПорядкаСтрок(<Действие>)
```

Англоязычный синоним:

```
OnChangeLinesOrder
```

Параметры:

<Действие>	Число: 1 — перемещение строки вниз; -1 (минус единица) — перемещение строки вверх.

Описание:

Вызов предопределенной процедуры ПриИзмененииПорядкаСтрок производится системой 1С:Предприятие неявно при интерактивной попытке перемещения строк вверх-вниз и перенумерации (до выполнения действия).

Если в данной предопределенной процедуре установлен статус возврата 0 (например, если данному пользователю нельзя устанавливать данное значение отбора проводок), установка отбора не будет выполнена.

Данная предопределенная процедура может располагаться только в модуле формы операции (см. «Виды программных модулей»).

Пример:

```
Процедура ПриИзмененииПорядкаСтрок(Направление)
   СтатусВозврата(0);
КонецПроцедуры
```

См. также: СтатусВозврата