1С:Предприятие
Версия 7.7
Описание встроенного языка
Часть 2

Содержание

СОДЕРЖАНИЕ 1

ГЛАВА 18 РАБОТА С ЖУРНАЛАМИ ОПЕРАЦИЙ 11

КОНТЕКСТ РАБОТЫ С ЖУРНАЛАМИ ОПЕРАЦИЙ 11
Методы контекста Модуля формы журнала операций 11
ВидыОтбора 11
ЗакладкиОтбора 12
УстановитьОтбор 13
ПолучитьОтбор 13
УстановитьИнтервал 13
НачалоИнтервала 14
КонецИнтервала 14
Предопределенные процедуры модуля формы журнала операций 14
ПриУстановкеОтбора 14
ПриУстановкеИнтервала 15

ГЛАВА 19 РАБОТА С ЖУРНАЛАМИ ПРОВОДОК 16

КОНТЕКСТ РАБОТЫ С ЖУРНАЛАМИ ПРОВОДОК 16
Методы контекста Модуля формы журнала проводок 16
ВидыОтбора 16
ЗакладкиОтбора 17
УстановитьОтбор 18
ПолучитьОтбор 18
УстановитьИнтервал 18
НачалоИнтервала 19
КонецИнтервала 19
Предопределенные процедуры модуля формы журнала проводок 19
ПриУстановкеОтбора 19
ПриУстановкеИнтервала 20
ПриПоказеПроводокПоДокументу 20

ГЛАВА 20 РАБОТА С БУХГАЛТЕРСКИМИ ИТОГАМИ 21

КОНТЕКСТ РАБОТЫ С БУХГАЛТЕРСКИМИ ИТОГАМИ 21
Общие свойства 21
ИспользоватьПланСчетов 21
ИспользоватьРазделительУчета 22
РАБОТА С ОСНОВНЫМИ ИТОГАМИ 22
Остатки и обороты по счетам 22
СНД, СНК, СКД, СКК, ДО, КО 22
Обороты между счетами 23
ОБ 23
Развернутое сальдо по субсчетам 23
СНДР, СНКР, СКДР, СККР 23
Развернутое сальдо по субконто 24
СНДРС, СНКРС, СКДРС, СККРС 24
Установка периода итогов 25
ПериодД 25
ПериодКВ 25
ПериодКВН 26
ПериодМ 26
ПериодМНК 26
ПериодМНГ 27
НачПериода 27
КонПериода 27
ОсновныеИтоги 27
РАБОТА С ВРЕМЕННЫМИ ИТОГАМИ 28
Рассчитать 28
Актуальность 28
РАБОТА В РЕЖИМЕ ЗАПРОСА 29
ВыполнитьЗапрос 29
ВключатьСубсчета 30
Опции 31
ИспользоватьСубконто 31
ИспользоватьКорСубконто 32
РАБОТА С РЕЗУЛЬТАТАМИ ЗАПРОСА 32

Методы выборки результатов запроса ... *32*
ВыбратьСчета ... 32
ПолучитьСчет ... 33
ВыбратьКорСчета ... 34
ПолучитьКорСчет ... 34
ВыбратьВалюты ... 35
ПолучитьВалюту ... 36
ВыбратьПериоды ... 36
ПолучитьПериод ... 37
ВыбратьСубконто ... 38
ПолучитьСубконто ... 39
ВыбратьКорСубконто ... 39
ПолучитьКорСубконто ... 40
Атрибуты для обращения к результатам запроса ... *41*
Счет ... 41
КорСчет ... 41
Валюта ... 41
НачДата ... 42
КонДата ... 42
Операция ... 42
Методы обращения к результатам запроса ... *43*
Субконто ... 43
КорСубконто ... 43
ПредставлениеСубконто ... 43
ПредставлениеКорСубконто ... 44
ЭтоГруппа ... 44
СНД, СНК, СКД, СКК, ДО, КО ... 45
СНДРС, СНКРС, СКДРС, СККРС ... 45
КорДО, КорКО ... 46
ВыбранаПоДт, ВыбранаПоКт ... 47

ГЛАВА 21 РАБОТА С КОРРЕКТНЫМИ ПРОВОДКАМИ ... 48

КОНТЕКСТ РАБОТЫ С ОБЪЕКТОМ «КОРРЕКТНЫЕПРОВОДКИ» ... 48
Атрибуты объекта «КорректныеПроводки» ... *48*
Комментарий ... 48
СчетДт ... 48
СчетКт ... 48
Методы объекта «КорректныеПроводки» ... *49*
Выбрана ... 49
ВыбратьКорректныеПроводки ... 49
ВыбратьКорректныеПроводкиПоСчету ... 49
ПолучитьКорректнуюПроводку ... 50
Новая ... 50
Записать ... 50
Удалить ... 50

ГЛАВА 22 РАБОТА С ЖУРНАЛАМИ РАСЧЕТОВ ... 52

КОНТЕКСТ РАБОТЫ С ЖУРНАЛОМ РАСЧЕТА ... 52
Контекст работы с записями журнала расчетов ... *52*
Контекст работы с периодом журнала расчетов ... *53*
Атрибуты периода журнала расчетов ... *53*
ДатаНачала ... 53
ДатаОкончания ... 53
ОписательПериода ... 54
Методы периода журнала расчетов ... *54*
ПрибавитьПериод ... 54
Атрибуты журнала расчетов ... *55*
Документ ... 55
РодительскийДокумент ... 55
Объект ... 55
ВидРасч ... 56
ДатаНачала ... 57
ДатаОкончания ... 57
ПериодДействия ... 57
ПериодРегистрации ... 58
Сторно ... 58
Рассчитана ... 59
Исправлена ... 59
Фиксирована ... 60
Перерасчет ... 60
ПервичнаяЗапись ... 61
Результат ... 61
<Реквизит> ... 61

Методы журнала расчетов ... *62*
НачалоТекущегоПериода ... 62
КонецТекущегоПериода ... 62
НачалоПериодаПоДате ... 62
КонецПериодаПоДате ... 63
ПериодПоДате ... 63
УстановитьТекущийПериод ... 63
ТекущийПериод ... 63
ПолучитьЗапись ... 64
ВыполнитьРасчет ... 64
ОписательПериода ... 65
ТекущаяЗапись ... 65
НайтиЗапись ... 65
ФиксироватьЗапись ... 66
ОсвободитьЗапись ... 66
ВвестиПерерасчет ... 66
ВвестиПерерасчетНаОсновании ... 67
Вид ... 68
ПредставлениеВида ... 68
НазначитьТип ... 68
УстановитьРеквизит ... 69
ВвестиРасчет ... 70
ВвестиРасчетНаОсновании ... 71
ЗаписатьРасчет ... 72
ЗаписатьРасчетНаОсновании ... 73
Рассчитать ... 73
ВыбратьЗаписи ... 74
ВыбратьПериод ... 74
ВыбратьЗаписиПоОбъекту ... 75
ВыбратьЗаписиПоДокументу ... 75
ВыбратьПериодПоОбъекту ... 76
ВыбратьПоЗначению ... 77
Новая ... 77
Записать ... 78
УдалитьЗапись ... 78
Исправить ... 79
ОтменитьИсправление ... 79
Методы контекста Модуля формы журнала расчетов ... *79*
ВидыОтбора ... 79
УстановитьОтбор ... 79
ПолучитьОтбор ... 80
ЗакладкиОтбора ... 81
ГраницаПросмотра ... 81
УстановитьПредставление ... 81
ПолучитьПредставление ... 82
РассчитыватьПриОтменеИсправления ... 83
Предопределенные процедуры Модуля формы журнала расчетов ... *83*
ПриИсправленииРезультата ... 83
ПриОтменеИсправления ... 84
ПриРасчете ... 84
ПриВыбореВладельца ... 85
ПриУстановкеОтбора ... 85
ПриУстановкеГраницыПросмотра ... 86
ПриУстановкеПредставления ... 86

ГЛАВА 23 РАБОТА С ВИДАМИ И ГРУППАМИ РАСЧЕТОВ ... 87

КОНТЕКСТ РАБОТЫ С ВИДАМИ РАСЧЕТОВ И ГРУППАМИ РАСЧЕТОВ ... 87
Атрибуты видов расчета и групп видов расчета ... *87*
Код ... 87
Наименование ... 87
Атрибуты видов расчета ... *88*
Очередность ... 88
ПриоритетВытеснения ... 88
Методы видов расчета ... *89*
ПолучитьАтрибут ... 89
ВходитВГруппу ... 89
Выбран ... 90
ВытесняетВидРасчета ... 90
ВытесняетсяВидомРасчета ... 91
Методы групп видов расчета ... *91*
СодержитВидРасчета ... 91
Количество ... 91
ПолучитьРасчет ... 91

ГЛАВА 24 РАБОТА С ПРАВИЛАМИ ПЕРЕРАСЧЕТА ... 93

КОНТЕКСТ РАБОТЫ С ПРАВИЛАМИ ПЕРЕРАСЧЕТА ... 93
Атрибуты правил перерасчета ... *93*
Тип ... 93
КоличествоПериодов ... 93
Методы правил перерасчета ... *94*
КоличествоВедущих ... 94
ИмеетВедущий ... 94
ПолучитьВедущий ... 94
ДобавитьКакВедущий ... 94
УдалитьВсеВедущие ... 95
КоличествоПодчиненных ... 95
ИмеетПодчиненный ... 95
ПолучитьПодчиненный ... 95
ДобавитьКакПодчиненный ... 96
УдалитьВсеПодчиненные ... 96
Применять ... 96

ГЛАВА 25 РАБОТА С КАЛЕНДАРЯМИ И ПРАЗДНИКАМИ ... 97

КОНТЕКСТ РАБОТЫ С КАЛЕНДАРЯМИ ... 97
КОНТЕКСТ РАБОТЫ С ПРАЗДНИКАМИ ... 97
Атрибуты и методы объекта Календари ... *98*
<ИмяКалендаря> ... 98
ПолучитьАтрибут ... 98
Атрибуты календарей и праздников ... *98*
Дата ... 98
Значение ... 98
Общие методы календарей и праздников ... *98*
ПолучитьАтрибут ... 98
УстановитьАтрибут ... 99
Методы календарей ... *99*
Выбран ... 99
ВыбратьДаты ... 99
СледующаяДата ... 100
Дней ... 100
Часов ... 101
Автозаполнение ... 101
УчитыватьПраздники ... 101
ПолучитьДату ... 102
Методы праздников ... *102*
Новый ... 102
Удалить ... 103
ВыбратьДаты ... 103
СледующаяДата ... 103

ГЛАВА 26 РАБОТА С ПОСЛЕДОВАТЕЛЬНОСТЯМИ ДОКУМЕНТОВ ... 104

КОНТЕКСТ РАБОТЫ С ПОСЛЕДОВАТЕЛЬНОСТЯМИ ... 104
Методы последовательностей ... *104*
ПолучитьПозицию ... 104
Получить ... 104
ПолучитьДокумент ... 104
ПолучитьДату ... 105
ПолучитьВремя ... 105
Установить ... 105
Сравнить ... 105
ПринадлежитПоследовательности ... 106
Проверить ... 106

ГЛАВА 27 РАБОТА С ОБЪЕКТОМ ПЕРИОДИЧЕСКИЙ ... 107

КОНТЕКСТ РАБОТЫ С ОБЪЕКТОМ ПЕРИОДИЧЕСКИЙ ... 107
Атрибуты объекта Периодический ... *107*
Значение ... 107
ДатаЗнач ... 107
Методы объекта Периодический ... *107*
ИспользоватьОбъект ... 107
НазначитьТип ... 108
ЗначениеНаДату ... 109
НайтиЗначение ... 109
ВыбратьЗначения ... 109
ВыбратьПоДокументу ... 110
ПолучитьЗначение ... 110
ОбратныйПорядок ... 111
ТекущийДокумент ... 111
ТекущийОбъект ... 112

ТекущийРеквизит ... 113
НомерСтроки ... 113
Записать ... 114
Удалить ... 114

ГЛАВА 28 РАБОТА СО СПИСКОМ ЗНАЧЕНИЙ ... 116

КОНТЕКСТ РАБОТЫ СО СПИСКОМ ЗНАЧЕНИЙ ... 116
Методы объекта Список Значений ... 116
ДобавитьЗначение ... 116
ВставитьЗначение ... 117
РазмерСписка ... 117
НайтиЗначение ... 117
ПолучитьЗначение ... 118
УстановитьЗначение ... 118
Получить ... 118
Установить ... 118
УдалитьЗначение ... 119
УдалитьВсе ... 119
Сортировать ... 119
СортироватьПоПредставлению ... 119
СдвинутьЗначение ... 120
Принадлежит ... 120
ВыбратьЗначение ... 120
ОтметитьЗначения ... 121
Пометка ... 122
ТекущаяСтрока ... 122
ИзСтрокиСРазделителями ... 122
ВСтрокуСРазделителями ... 123
Выгрузить ... 123

ГЛАВА 29 РАБОТА С ТАБЛИЦЕЙ ЗНАЧЕНИЙ ... 124

КОНТЕКСТ РАБОТЫ С ТАБЛИЦЕЙ ЗНАЧЕНИЙ ... 124
Атрибуты Таблицы Значений ... 124
НомерСтроки ... 124
<ИдентификаторКолонки> ... 124
Методы объекта Таблица Значений ... 125
КоличествоКолонок ... 125
НоваяКолонка ... 125
ВставитьКолонку ... 125
УдалитьКолонку ... 126
УстановитьПараметрыКолонки ... 126
ПолучитьПараметрыКолонки ... 126
КоличествоСтрок ... 127
НоваяСтрока ... 127
УдалитьСтроку ... 127
УдалитьСтроки ... 127
ВыбратьСтроки ... 128
ПолучитьСтроку ... 128
ВыбратьСтроку ... 128
ПолучитьСтрокуПоНомеру ... 128
СдвинутьСтроку ... 129
УстановитьЗначение ... 129
ПолучитьЗначение ... 129
НайтиЗначение ... 129
Сортировать ... 130
Очистить ... 130
Итог ... 130
Заполнить ... 130
Свернуть ... 131
Выгрузить ... 131
Загрузить ... 131
ВидимостьКолонки ... 132
ТекущаяСтрока ... 132
ТекущаяКолонка ... 132
Фиксировать ... 133
ВыводитьПиктограммы ... 133
Пример использования объекта Таблица Значений ... 133

ГЛАВА 30 АТРИБУТЫ И МЕТОДЫ КОНТЕКСТА МОДУЛЯ ФОРМЫ ... 135

Атрибуты контекста Модуля формы ... 135
СтрокаДействийФормы ... 135
Форма ... 135
Атрибуты объекта Форма ... 135
Закладки ... 135

<ЭлементДиалога> 136
Параметр 136
Методы объекта Форма *136*
ТолькоПросмотр 136
Обновить 137
ИспользоватьЗакладки 137
ИспользоватьСлой 137
Заголовок 138
ПанельИнструментов 138
КнопкаПоУмолчанию 138
ОбработкаВыбораСтроки 138
ВыполнитьВыбор 139
РежимВыбора 139
МодальныйРежим 139
ПолучитьАтрибут 139
АктивныйЭлемент 140
ТекущаяКолонка 140
Закрыть 140
Методы элементов диалога *140*
Видимость 140
Доступность 141
Редактирование 141
Цвет 141
Маска 142
ВыборГруппы 142
ВыполнятьФормулуТолькоПриИзменении 143
Заголовок 143
УстановитьТип 143
НазначитьТип 143
НеИзменятьВид 144
Методы контекста Модуля формы *144*
ОткрытьПодбор 144
УстановитьЗначениеВПодборе 145
ПолучитьЗначениеИзПодбора 146
Активизировать 146
АктивизироватьОбъект 146
Предопределенные процедуры Модуля формы *147*
ПриОткрытии 147
ПриПовторномОткрытии 147
ПриЗакрытии 147
ПриВыбореЗакладки 148
ПриНачалеВыбораЗначения 148
ОбработкаПодбора 149
ОбработкаВыбораЗначения 149
ПриВыбореСтроки 149
Атрибуты и методы контекста Модуля формы отчета (обработки) *150*
Таблица 150
<ИмяОбласти> 150
РасположениеФайла 151
Предопределенные процедуры модуля формы отчета (обработки) *151*
ВводНового 151
ПриОткрытии 152

ГЛАВА 31 РАБОТА С ТАБЛИЦАМИ 153

КОНТЕКСТ РАБОТЫ С ТАБЛИЦАМИ 155
Атрибуты таблиц *155*
ТекущийОбъект 155
Методы таблиц *156*
ИсходнаяТаблица 156
ИспользоватьФормат 156
Открыть 156
Вывести 157
ПолучитьСекцию 157
ВывестиСекцию 157
ПрисоединитьСекцию 158
НоваяСтраница 159
НоваяКолонка 159
ШиринаТаблицы 159
ВысотаТаблицы 159
ШиринаСекции 160
ВысотаСекции 160
ТолькоПросмотр 160
Очистить 160
Показать 161

Защита ... 161
Записать ... 162
ЗначениеТекущейЯчейки ... 162
Область ... 162
ПовторятьПриПечатиСтроки ... 163
ПовторятьПриПечатиСтолбцы ... 163
Опции ... 163
ОбластьПечати ... 164
ПараметрыСтраницы ... 164
КоличествоЭкземпляров ... 165
ЭкземпляровНаСтранице ... 165
Напечатать ... 165
Атрибуты и методы объекта «СекцияТаблицы» ... *166*
<ИмяОбласти> ... 166
Область ... 166
Атрибуты и методы объекта «ОбластьТаблицы» ... *166*
Текст ... 166
Расшифровка ... 167
Объединить ... 167
Шрифт ... 167
РазмерШрифта ... 167
Полужирный ... 168
Курсив ... 168
Подчеркнутый ... 168
ВертикальноеПоложение ... 169
ГоризонтальноеПоложение ... 169
Контроль ... 169
РамкаСверху ... 170
РамкаСнизу ... 170
РамкаСлева ... 170
РамкаСправа ... 171
Рамка ... 171
РамкаОбвести ... 172
ЦветФона ... 172
ЦветРамки ... 172
ЦветТекста ... 173
ВысотаСтроки ... 173
ШиринаСтолбца ... 174
Атрибуты и методы таблицы в режиме ввода данных ... *174*
<ИмяОбласти> ... 174
АктивнаяОбласть ... 174
Выгрузить ... 175
Загрузить ... 175
Атрибуты и методы области таблицы в режиме ввода данных ... *175*
Значение ... 175
Формат ... 176
УстановитьТип ... 176
НазначитьТип ... 176
Доступность ... 176
Редактирование ... 177
Системные предопределенные процедуры работы с таблицами ... *177*
ОбработкаЯчейкиТаблицы ... 177
ПриВыбореЯчейкиТаблицы ... 178

ГЛАВА 32 РАБОТА С ТЕКСТОМ ... 179

КОНТЕКСТ РАБОТЫ С ТЕКСТОМ ... 179
Методы текста ... *179*
КоличествоСтрок ... 179
ПолучитьСтроку ... 180
Открыть ... 180
Шаблон ... 180
ФиксШаблон ... 181
ВставитьСтроку ... 181
ДобавитьСтроку ... 181
ЗаменитьСтроку ... 182
УдалитьСтроку ... 182
ТолькоПросмотр ... 182
Показать ... 182
Очистить ... 183
КодоваяСтраница ... 183
Записать ... 183

ГЛАВА 33 РАБОТА С ЗАПРОСАМИ ... 184

КОНТЕКСТ РАБОТЫ С ЗАПРОСАМИ ... 184

Структура запросов и методика их использования ... 184
Атрибуты запросов ... *186*
Методы запросов ... *186*
Выполнить ... 186
ИспользоватьГрафуОтбора ... 187
Группировка ... 188
ЭтоГруппа ... 189
НачалоПериода ... 189
КонецПериода ... 190
Получить ... 190
ВНачалоВыборки ... 190
Выгрузить ... 191
ЗначениеУпорядочивания ... 191
ПолучитьАтрибут ... 192

ГЛАВА 34 ЯЗЫК ЗАПРОСОВ ... 193

Формат текста описания запроса ... 193
Соглашения и обозначения ... *193*
Зарезервированные слова языка запросов ... *193*
Комментарий ... *194*
Константы и переменные запросов ... 194
Константы ... *194*
Внутренние переменные ... *194*
Конкретизация переменной ... *195*
Внешние переменные ... *195*
Атрибуты, доступные при описании внутренних переменных ... 196
Операторы языка запросов ... 197
Объявление внутренней переменной ... *197*
Период С ... 199
ОбрабатыватьДокументы ... 201
ОбрабатыватьОперации ... 201
Обрабатывать ... 202
Функция ... 202
Группировка ... 204
Без итогов ... 207
Условие ... 207
Примеры использования Запросов ... 209
Печать каталога товаров ... *209*
Отчет по неходовым товарам ... *209*
Отчет по регистру с точностью до строки документа ... *211*
Анализ счета ... *211*
Разработка вложенных отчетов ... *212*
Способы оптимизации формирования отчетов ... 214

ГЛАВА 35 РАБОТА С КАРТИНКАМИ ... 216

Контекст работы с картинками ... 216
Методы объекта Картинка ... *216*
Загрузить ... 216
Сохранить ... 216
РежимРисования ... 216
УстановитьКартинку ... 217

ГЛАВА 36 РАБОТА С ДИАГРАММАМИ ... 218

Контекст работы с диаграммами ... 218
Объект «Диаграмма» — основные принципы и понятия, используемые при визуальной настройке и управлении ... 218
Атрибуты диаграммы ... *219*
Заголовок ... 219
Методы диаграммы ... *219*
КоличествоСерий ... 219
КоличествоТочек ... 219
УстановитьИмяСерии ... 219
УстановитьИмяТочки ... 220
ЦветСерии ... 220
АвтоУстановкаИменСерий ... 220
АвтоУстановкаИменТочек ... 220
УстановитьЗначение ... 220
Обновление ... 221
Очистить ... 221
Пример использования ... 221

ГЛАВА 37 РАБОТА С ФАЙЛАМИ ... 223

Контекст работы с Файлами ... 223
Методы объекта «ФС» ... *223*
ВыбратьФайл ... 223
ВыбратьФайлКартинки ... 224
ВыбратьКаталог ... 224
СуществуетФайл ... 225
КопироватьФайл ... 225
УдалитьФайл ... 225
ПереименоватьФайл ... 225
НайтиПервыйФайл ... 226
НайтиСледующийФайл ... 226
АтрибутыФайла ... 226
СоздатьКаталог ... 227
УдалитьКаталог ... 227
УстТекКаталог ... 227
ТекКаталог ... 228
WindowsКаталог ... 228
СвободноеМестоНаДиске ... 228

ГЛАВА 38 РАБОТА С БАЗАМИ ДАННЫХ ФОРМАТА DBF ... 229

Основные понятия ... 229
Поля и записи ... 229
Таблица, структура таблицы, файл базы данных ... 229
Индексы, выражения индекса и фильтра ... 229
Индексный файл ... 229
Назначение агрегатного типа данных XBase ... 229
Атрибуты объекта и поля базы данных ... 229
Запись изменений в базу данных ... 229
Работа с индексными файлами ... 230
Удаление записей ... 230
Создание базы данных, индекса, индексного файла ... 230
Ограничения ... 230
Контекст работы с XBase ... 230
Атрибуты объекта XBase ... *231*
<Поле> ... 231
Ключ ... 231
Методы объекта XBase ... *231*
СоздатьФайл ... 231
ОткрытьФайл ... 231
Открыта ... 232
ЗакрытьФайл ... 232
ОчиститьФайл ... 232
Сжать ... 232
Переиндексировать ... 233
ПоказыватьУдаленные ... 233
Первая ... 233
Последняя ... 234
Следующая ... 234
Предыдущая ... 234
НомерЗаписи ... 235
Перейти ... 235
ВКонце ... 236
ВНачале ... 236
ТекущийИндекс ... 236
Найти ... 237
НайтиПоКлючу ... 237
ПолучитьЗначениеПоля ... 237
УстановитьЗначениеПоля ... 238
Добавить ... 238
Скопировать ... 238
Автосохранение ... 238
Записать ... 239
Отменить ... 239
Удалить ... 239
ЗаписьУдалена ... 240
Восстановить ... 240
Очистить ... 240
КоличествоЗаписей ... 241
КоличествоПолей ... 241
КоличествоИндексов ... 241
ОписаниеПоля ... 242
ОписаниеИндекса ... 242
НомерПоля ... 242
ДобавитьПоле ... 243

ДобавитьИндекс ... 243
СоздатьИндексныйФайл ... 243
КодоваяСтраница ... 244
КодОшибки ... 244
Выражение и фильтр индекса ... 245
Функции, применяемые в выражениях ... *246*
ГЛАВА 39 РАБОТА С МЕТАДАННЫМИ ... 247
Контекст работы с метаданными ... 247
Атрибуты и методы объекта «Метаданные» ... 247
Методы работы с метаданными ... *247*
Выбран ... 248
Родитель ... 248
ПолныйИдентификатор ... 248
Представление ... 248
ДлинаПредставленияЗначения ... 248
ГЛАВА 40 СВЯЗЬ С ВНЕШНИМИ ПРИЛОЖЕНИЯМИ ПОСРЕДСТВОМ МЕХАНИЗМОВ DDE И OLE AUTOMATION ... 250
Контекст работы с внешними приложениями ... 250
Методы внешних приложений ... 250
Работа системы 1С:Предприятие в качестве OLE Automation сервера ... 251
Атрибуты системы 1С:Предприятие как OLE Automation сервера ... *252*
Методы системы 1С:Предприятие как OLE Automation сервера ... *252*
Initialize ... 252
EvalExpr ... 252
CreateObject ... 253
ExecuteBatch ... 253
Работа системы 1С:Предприятие в качестве DDE сервера ... *254*

Глава 18
Работа с Журналами операций

Контекст работы с журналами операций

Журнал операций — средство для работы со списком операций. В терминах языка журнал операций не является специальным типом данных (он не имеет значения, его нельзя создать при помощи функции `СоздатьОбъект`).

С журналом в системе связана форма отображения списка операций и программный модуль формы журнала операций (см. «Виды программных модулей»). В локальном контексте этого программного модуля непосредственно доступны реквизиты формы. Кроме того, здесь непосредственно доступен объект «Операция», содержащий значение выбранной в журнале операции. Другими словами, в модуле формы журнала операций обращение к атрибутам и методам текущей операции выполняется напрямую.

Методы контекста Модуля формы журнала операций

Описанные в данном разделе методы доступны только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

ВидыОтбора

Установить доступные виды отборов для журнала операций.

Синтаксис:

`ВидыОтбора(<СписокОтборов>)`

Англоязычный синоним:

`KindsOfSelection`

Параметры:

`<СписокОтборов>`	Необязательный параметр. Строковое выражение. Может принимать значения: список имен отборов (через запятую) — в журнале операций будут доступны только указанные виды отборов; символ «*» — для журнала используются все назначенные в Конфигураторе виды отборов; пустая строка ("") — запрещаются все виды отборов. Если параметр не указан, метод возвращает текущий список разрешенных отборов.

Возвращаемое значение:

Строковое значение, содержащее текущий (на момент до вызова метода) список отборов для журнала операций. Имена отбора в возвращаемой строке разделяются запятыми.

Описание:

Метод `ВидыОтбора` устанавливает доступные виды отборов журнала операций. Использование данного метода влияет на полноту списка видов отбора, который выдается пользователю в диалоге «Отбор» при работе с системой 1С:Предприятие.

Метод `ВидыОтбора` позволяет ограничить число видов отбора, доступных конкретному пользователю при работе с журналом операций, или совсем запретить выполнение отбора в журнале. Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Можно выделить 4 варианта использования данного метода.

1. Если параметр `<СписокОтборов>` не указан, метод возвращает текущий список отборов, установленных для журнала операций.

2. Чтобы ограничить использование отборов какими-либо конкретными видами отбора, список этих видов отбора следует передать методу в качестве параметра. Указывать в списке можно как имена отборов, автоматически задаваемые системой 1С:Предприятие, так и отборы, созданные в процессе конфигурирования: по реквизитам операции и по общим реквизитам документов.

«Системные» отборы задаются включением соответствующих опций в окне редактирования свойств операции в конфигураторе. Каждому виду отбора соответствует «системное» имя отбора — то, которое выдается пользователю в диалоге «Отбор», и которое можно использовать во встроенном языке (в том числе и при вызове метода `ВидыОтбора`).

Для журнала операций допустимы следующие «системные» имена отборов:

`СуммаОперации` — отбор по сумме операции;

`Содержание` — отбор по содержанию операции.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по дополнительным реквизитам операции. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств дополнительного реквизита операции. В этих случаях в качестве имен отбора следует использовать идентификаторы дополнительных реквизитов операции.

Кроме того в качестве имен отборов могут выступать идентификаторы граф отбора, в которых участвуют бухгалтерские документы или виды субконто, или данные операции или проводки.

Также в жунале операций возможен отбор по видам документов, для которых установлен признак «Бухгалтерский учет». Имена этих отборов определяются идентификаторами видов документов.

3. Если параметр <СписокОтборов> равен «*», разрешаются все виды отборов, установленные для журнала операций в конфигураторе. Вызов метода ВидыОтбора с таким параметром позволяет отключить ранее установленное ограничение на использование видов отборов.

4. Если в качестве параметра методу ВидыОтбора передана пустая строка, метод запрещает пользователю выполнение любых отборов в журнале операций.

Замечание. Ограничение списка отборов, выполненное при помощи метода ВидыОтбора, не исключает использование «запрещенного» отбора средствами встроенного языка. Например, методом ВидыОтбора("") можно запретить использование любых отборов в журнале операций, но тутже из языка установить отбор операций по конкретному содержанию операции. В этом случае пользователь будет просматривать операции с конкретным содержанием, но не будет иметь возможность отобрать операции с другим содержанием.

Пример:

```
ВидыОтбора("Клиент, Автор");
```

ЗакладкиОтбора

Установить в форме журнала операций закладки для интерактивного выбора значения отбора.

Синтаксис:

```
ЗакладкиОтбора(<ИмяОтбора>, <ВИнтервалеЖурнала>, <УстановитьНаЗначение>,
                            <ЗначениеОтбора>)
```

Англоязычный синоним:

TabCtrlSelection

Параметры:

<ИмяОтбора>	Строковое выражение: имя отбора.
<ВИнтервалеЖурнала>	ловое выражение: признак отбора только в интервале журнала. Может принимать значения: 1 — текущие значения отбора для закладок выбираются только по проводкам в установленном интервале журнала; 0 — текущие значения отбора для закладок выбираются по всем проводкам журнала. По умолчанию — 0.
<УстановитьНаЗначение>	Числовое выражение: признак выбора значения отбора для показа. Параметр может принимать значения: 1 — для показа выбирается отбор по значению, указанному в параметре <ЗначениеОтбора>; 0 — текущее значение отображаемой закладки отбора устанавливается на первое существующее значение. По умолчанию — 0.
<ЗначениеОтбора>	Значение отбора.

Описание:

Метод ЗакладкиОтбора устанавливает в форме журнала операций закладки для интерактивного осуществления отбора. При включении закладок в верхней части журнала появляются ярлыки, соответствующие значениям отбора. Щелчком мыши по ярлыку можно открыть «страницу» журнала операций: такая «страница» будет содержать операции, отобранные по указанному значению.

Для включения отбора необходимо в качестве параметра <ИмяОтбора> передать методу имя отбора. Можно использовать как имена отборов, автоматически задаваемые системой 1С:Предприятие, так и отборы, созданные в процессе конфигурирования: по реквизитам операции и по общим реквизитам документов.

«Системные» отборы задаются включением соответствующих опций в окне редактирования свойств операции в конфигураторе. Каждому виду отбора соответствует «системное» имя отбора — то, которое выдается пользователю в диалоге «Отбор», и которое можно использовать во встроенном языке.

Для журнала операций допустимы следующие «системные» имена отборов:

СуммаОперации — отбор по сумме операции;

Содержание — отбор по содержанию операции.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по дополнительным реквизитам операции. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств дополнительного реквизита операции. В этих случаях в качестве имен отбора следует использовать идентификаторы дополнительных реквизитов операции.

Кроме того в качестве имен отборов могут выступать идентификаторы граф отбора, в которых участвуют бухгалтерские документы или виды субконто, или данные операции или проводки.

Также в жунале операций возможен отбор по видам документов, для которых установлен признак «Бухгалтерский учет». Имена этих отборов определяются идентификаторами видов документов.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
ЗакладкиОтбора("Автор", 1, , );
```

УстановитьОтбор

Установить отбор журнала операций.

Синтаксис:

```
УстановитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)
```

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 операций. Для установки отбора методу должны быть переданы 2 параметра: имя отбора и значение отбора.

Возможные виды отбора для журнала операций задаются в конфигураторе. В окне редактирования свойств операции можно включить опции, соответствующие различным видам отбора. Для таких отборов система 1С:Предприятие использует зарезервированные имена:

СуммаОперации — отбор по сумме операции;

Содержание — отбор по содержанию операции.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по дополнительным реквизитам операции и общим реквизитам документов. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств, соответственно, дополнительного реквизита операции или общего реквизита документа. В этих случаях в качестве имен отбора следует использовать идентификаторы дополнительных реквизитов операции или общих реквизитов документов.

Кроме того в качестве имен отборов могут выступать идентификаторы граф отбора, в которых участвуют бухгалтерские документы или виды субконто, или данные операции или проводки.

Также в жунале операций возможен отбор по видам документов, для которых установлен признак «Бухгалтерский учет». Имена этих отборов определяются идентификаторами видов документов. Эти оборы не имеют значения отбора.

Отбор устанавливается по значению, указанному в параметре <ЗначениеОтбора>.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
Изм = УстановитьОтбор("Склады", Склад!);
```

ПолучитьОтбор

Возвратить текущее значение отбора журнала операций.

Синтаксис:

```
ПолучитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)
```

Англоязычный синоним:

```
GetSelection
```

Параметры:

<ИмяОтбора>	Необязательный параметр. Имя переменной, куда будет записано строковое значение имени отбора.
<ЗначениеОтбора>	Необязательный параметр. Имя переменной, куда будет записано значение отбора.

Возвращаемое значение:

Число: 1 — отбор включен; 0 — отбор не включен.

Описание:

Метод ПолучитьОтбор позволяет получить информацию о текущем состоянии отбора журнала операций. Если отбор в журнале операций включен, метод возвращает 1 и записывает имя отбора и текущее значение отбора в переменные, передаваемые методу ПолучитьОтбор в качестве параметров.

Возвращаемое значение метода может использоваться для определения того, включен отбор в журнале операций или нет.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
Изм = ПолучитьОтбор();
```

УстановитьИнтервал

Установить интервал журнала операций.

Синтаксис:

```
УстановитьИнтервал(<ДатаНач>, <ДатаКон>, <ФлагИзменения>)
```

Англоязычный синоним:

```
SetRange
```

Параметры:

<ДатаНач>	Выражение типа «дата» — начальная дата интервала журнала операций.
<ДатаКон>	Выражение типа «дата» — конечная дата интервала журнала операций.
<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 УстановитьИнтервал позволяет установить интервал видимости записей в журнале операций. При работе с журналом будут доступны операции, даты которых лежат внутри указанного интервала.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
УстановитьИнтервал(НИ, КИ);
```

НачалоИнтервала

Возвратить дату начала интервала журнала операций.

Синтаксис:

```
НачалоИнтервала()
```

Англоязычный синоним:

```
BegOfRange
```

Возвращаемое значение:

Значение типа «Дата» — начальная дата интервала журнала операций.

Описание:

Метод НачалоИнтервала позволяет получить текущую начальную дату интервала журнала операций.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
НИ = НачалоИнтервала();
```

КонецИнтервала

Возвратить дату конца интервала журнала операций.

Синтаксис:

```
КонецИнтервала()
```

Англоязычный синоним:

```
EndOfRange
```

Возвращаемое значение:

Значение типа «Дата» — конечная дата интервала журнала операций.

Описание:

Метод КонецИнтервала позволяет получить конечную дату интервала журнала операций.

Данный метод доступен только в контексте модуля формы журнала операций (см. «Виды программных модулей»).

Пример:

```
КИ = КонецИнтервала();
```

Предопределенные процедуры модуля формы журнала операций

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в модулях форм журнала операций (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 в журнале операций.

Синтаксис:

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

Англоязычный синоним:

```
OnSetSelectInJournal
```

Параметры:

<ИмяОтбора>	Строковое выражение — имя отбора.
<ЗначениеОтбора>	Значение отбора.

Описание:

Вызов предопределенной процедуры ПриУстановкеОтбора производится системой 1С:Предприятие неявно при интерактивной попытке установить отбор в журнале операций.

Если в данной предопределенной процедуре установлен статус возврата 0 (например, если данному пользователю нельзя устанавливать данное значение отбора операций), установка отбора не будет выполнена.

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

Пример:

```
Процедура ПриУстановкеОтбора(ИмяОтбора, ЗначОтбора)
   Если НазваниеНабораПрав() = "Оператор" Тогда
      Если (ИмяОтбора = "Автор") И (ЗначОтбораоТекПольз) Тогда
```

```
            Предупреждение("У вас нет права просматривать чужие Операции!", 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);
   КюнецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 19
Работа с Журналами проводок

Контекст работы с журналами проводок

Журнал проводок — средство для работы со списком проводок. В терминах языка журнал проводок не является специальным типом данных (он не имеет значения, его нельзя создать при помощи функции `СоздатьОбъект`).

С журналом в системе связана форма отображения списка проводок и программный модуль формы журнала проводок (см. «Виды программных модулей»). В локальном контексте этого программного модуля непосредственно доступны реквизиты формы. Кроме того, здесь непосредственно доступен объект «Операция», содержащий значение операции, которой принадлежит выбранная в журнале проводка. Другими словами, в модуле формы журнала проводок обращение к атрибутам и методам текущей операции выполняется напрямую.

Методы контекста Модуля формы журнала проводок

Описанные в данном разделе методы доступны только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

ВидыОтбора

Установить доступные виды отборов для журнала проводок.

Синтаксис:

`ВидыОтбора(<СписокОтборов>)`

Англоязычный синоним:

`KindsOfSelection`

Параметры:

`<СписокОтборов>`	Необязательный параметр. Строковое выражение. Может принимать значения: список имен отборов через запятую — в журнале проводок будут доступны только указанные виды отборов; символ «*» — для журнала используются все назначенные в Конфигураторе виды отборов; пустая строка "" — запрещаются все виды отборов. Если параметр не указан, метод возвращает текущий список разрешенных отборов.

Возвращаемое значение:

Строковое значение, содержащее текущий (на момент до вызова метода) список отборов для журнала. Имена отбора в возвращаемой строке разделяются запятыми.

Описание:

Метод `ВидыОтбора` устанавливает доступные виды отборов журнала проводок. Использование данного метода влияет на полноту списка видов отбора, который выдается пользователю в диалоге «Отбор» при работе с системой 1С:Предприятие.

Метод `ВидыОтбора` позволяет ограничить число видов отбора, доступных конкретному пользователю при работе с журналом проводок, или совсем запретить выполнение отбора в журнале проводок. Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Можно выделить 4 варианта использования данного метода.

1. Если параметр `<СписокОтборов>` не указан, метод возвращает текущий список отборов, установленных для журнала проводок.

2. Чтобы ограничить использование отборов какими-либо конкретными видами отбора, список этих видов отбора следует передать методу в качестве параметра. Указывать в списке можно как имена отборов, автоматически задаваемые системой 1С:Предприятие, так и отборы, созданные в процессе конфигурирования: по реквизитам проводки и по субконто проводки.

«Системные» отборы задаются включением соответствующих опций в окне редактирования свойств проводки в конфигураторе. Каждому виду отбора соответствует «системное» имя отбора — то, которое выдается пользователю в диалоге «Отбор», и которое можно использовать во встроенном языке (в том числе, и при вызове метода ВидыОтбора).

Для журнала проводок допустимы следующие «системные» имена отборов:

`Сумма` — отбор по сумме проводки;

`ВалСумма` — отбор по валютной сумме проводки;

`Валюта` — отбор по валюте проводки;

`Количество` — отбор по количеству проводки;

`Счет` — отбор по счетам проводки;

`СчетДт` — отбор по счетам дебета проводки;

`СчетКт` — отбор по счетам кредита проводки;

`ПланСчетов` — отбор по плану счетов.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по субконто проводки и дополнительным реквизитам проводки. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств вида субконто или дополнительного реквизита проводки. В этих случаях в качестве имен отбора следует использовать идентификаторы видов субконто или дополнительных реквизитов проводки.

3. Если параметр <СписокОтборов> равен «*», разрешаются все виды отборов, установленные для журнала проводок в конфигураторе. Вызов метода ВидыОтбора с таким параметром позволяет отключить ранее установленное ограничение на использование видов отборов.

4. Если в качестве параметра методу ВидыОтбора передана пустая строка, метод запрещает пользователю выполнение любых отборов в журнале проводок.

Замечание. Ограничение списка отборов, выполненное при помощи метода ВидыОтбора, не исключает использование «запрещенного» отбора средствами встроенного языка. Например, методом ВидыОтбора("") можно запретить использование любых отборов в журнале проводок, но установить отбор проводок по конкретной организации. В этом случае пользователь будет просматривать проводки, относящиеся только к указанной организации, но не будет иметь возможность изменить организацию.

Пример:

```
ВидыОтбора("Склад, Клиент, Автор");
```

ЗакладкиОтбора

Установить в форме журнала проводок закладки для интерактивного отбора.

Синтаксис:

```
ЗакладкиОтбора(<ИмяОтбора>, <ВИнтервалеЖурнала>, <УстановитьНаЗначение>,
               <ЗначениеОтбора>)
```

Англоязычный синоним:

```
TabCtrlSelection
```

Параметры:

<ИмяОтбора>	Строковое выражение: имя отбора.
<ВИнтервалеЖурнала>	Числовое выражение: признак отбора только в интервале журнала. Может принимать значения: 1 — текущие значения отбора для закладок выбираются только по проводкам в установленном интервале журнала; 0 — текущие значения отбора для закладок выбираются по всем проводкам журнала. По умолчанию — 0.
<УстановитьНаЗначение>	Числовое выражение: признак выбора значения отбора для показа. Параметр может принимать значения: 1 — для показа выбирается отбор по значению, указанному в параметре <ЗначениеОтбора>; 0 — текущее значение отображаемой закладки отбора устанавливается на первое существующее значение. По умолчанию — 0.
<ЗначениеОтбора>	Значение отбора.

Описание:

Метод ЗакладкиОтбора устанавливает в форме журнала проводок закладки для интерактивного осуществления отбора. При включении закладок в верхней части журнала проводок появляются ярлыки, соответствующие значениям отбора. Щелчком мыши по ярлыку можно открыть «страницу» журнала проводок: такая «страница» будет содержать проводки, отобранные по указанному значению.

Для включения отбора необходимо в качестве параметра <ИмяОтбора> передать методу имя отбора. Можно использовать как имена отборов, автоматически задаваемые системой 1С:Предприятие, так и отборы, созданные в процессе конфигурирования: по реквизитам проводки и по субконто проводки.

«Системные» отборы задаются включением соответствующих опций в окне редактирования свойств проводки в конфигураторе. Каждому виду отбора соответствует «системное» имя отбора — то, которое выдается пользователю в диалоге «Отбор», и которое можно использовать во встроенном языке.

Для журнала проводок допустимы следующие «системные» имена отборов:

Сумма — отбор по сумме проводки;

ВалСумма — отбор по валютной сумме проводки;

Валюта — отбор по валюте проводки;

Количество — отбор по количеству проводки;

Счет — отбор по счетам проводки;

СчетДт — отбор по счетам дебета проводки;

СчетКт — отбор по счетам кредита проводки;

ПланСчетов — отбор по плану счетов.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по субконто проводки и дополнительным реквизитам проводки. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств вида субконто или дополнительного реквизита проводки. В этих случаях в качестве имен отбора следует использовать идентификаторы видов субконто или дополнительных реквизитов проводки.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
ЗакладкиОтбора("Склады", 1, 1, Константа.ОснСклад);
```

УстановитьОтбор

Установить отбор журнала проводок.

Синтаксис:

```
УстановитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)
```

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 проводок. Для установки отбора методу должны быть переданы 2 параметра: имя отбора и значение отбора.

Возможные виды отбора для журнала проводок задаются в конфигураторе. В окне редактирования свойств проводки можно включить опции, соответствующие различным видам отбора. Для таких отборов система 1С:Предприятие использует зарезервированные имена:

`Сумма` — отбор по сумме проводки;

`ВалСумма` — отбор по валютной сумме проводки;

`Валюта` — отбор по валюте проводки;

`Количество` — отбор по количеству проводки;

`Счет` — отбор по счетам проводки;

`СчетДт` — отбор по счетам дебета проводки;

`СчетКт` — отбор по счетам кредита проводки;

`ПланСчетов` — отбор по плану счетов.

Кроме этого, в конфигураторе можно включить возможность выполнять отборы по субконто проводки и дополнительным реквизитам проводки. Для этого используется опция «Отбор» в закладке «Дополнительные» палитры свойств вида субконто или дополнительного реквизита проводки. В этих случаях в качестве имен отбора следует использовать идентификаторы видов субконто или дополнительных реквизитов проводки.

Отбор устанавливается по значению, указанному в параметре `<ЗначениеОтбора>`.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
Изм = УстановитьОтбор("Склады", Склад1);
```

ПолучитьОтбор

Возвратить текущее значение отбора журнала проводок.

Синтаксис:

```
ПолучитьОтбор(<ИмяОтбора>, <ЗначениеОтбора>)
```

Англоязычный синоним:

```
GetSelection
```

Параметры:

`<ИмяОтбора>`	Необязательный параметр. Имя переменной, куда будет записано строковое значение имени отбора.
`<ЗначениеОтбора>`	Необязательный параметр. Имя переменной, куда будет записано значение отбора.

Возвращаемое значение:

Числовое значение: 1 — отбор включен; 0 — отбор не включен.

Описание:

Метод `ПолучитьОтбор` возвращает текущее значение отбора журнала. Имя отбора и текущее значение отбора записываются в переменные, передаваемые методу `ПолучитьОтбор` в качестве параметров.

Возвращаемое значение метода может использоваться для определения того, включен отбор в журнале проводок или нет.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
Изм = ПолучитьОтбор();
```

УстановитьИнтервал

Установить интервал журнала проводок.

Синтаксис:

```
УстановитьИнтервал(<ДатаНач>, <ДатаКон>, <ФлагИзменения>}
```

Англоязычный синоним:

```
SetRange
```

Параметры:

`<ДатаНач>`	Выражение типа «дата» — начальная дата интервала журнала проводок.
`<ДатаКон>`	Выражение типа «дата» — конечная дата интервала журнала проводок.
`<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 УстановитьИнтервал позволяет установить интервал видимости записей в журнале проводок. При работе с журналом проводок будут доступны проводки, даты которых лежат внутри указанного интервала.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
УстановитьИнтервал(НИ, КИ);
```

НачалоИнтервала

Возвратить дату начала интервала журнала проводок.

Синтаксис:

```
НачалоИнтервала()
```

Англоязычный синоним:

```
BegOfRange
```

Возвращаемое значение:

Значение типа «Дата» — начальная дата интервала журнала проводок.

Описание:

Метод НачалоИнтервала позволяет получить начальную дату интервала журнала проводок.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
НИ = НачалоИнтервала();
```

КонецИнтервала

Возвратить дату конца интервала журнала проводок.

Синтаксис:

```
КонецИнтервала()
```

Англоязычный синоним:

```
EndOfRange
```

Возвращаемое значение:

Значение типа «Дата» — конечная дата интервала журнала проводок.

Описание:

Метод КонецИнтервала позволяет получить конечную дату интервала журнала проводок.

Данный метод доступен только в контексте модуля формы журнала проводок (см. «Виды программных модулей»).

Пример:

```
КИ = КонецИнтервала();
```

Предопределенные процедуры модуля формы журнала проводок

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в модулях форм журнала проводок (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 в журнале проводок.

Синтаксис:

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

Англоязычный синоним:

```
OnSetSelectInJournal
```

Параметры:

<ИмяОтбора>	Строковое выражение — имя отбора.
<ЗначениеОтбора>	Значение отбора.

Описание:

Вызов предопределенной процедуры ПриУстановкеОтбора производится системой 1С:Предприятие неявно при интерактивной попытке установить отбор в журнале проводок.

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 в модуле формы журнала проводок и в глобальном программном модуле. Если данная процедура присутствует в модуле формы журнала проводок, то будет вызвана она, если нет, то будет вызвана процедура из глобального модуля.

Пример:

```
Процедура ПриУстановкеОтбора(ИмяОтбора, ЗначОтбора)
   Если (ИмяОтбора = "Счет") И (ЗначОтбора.Валютный = 1) Тогда
```

```
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```
См. также: СтатусВозврата

ПриУстановкеИнтервала

Предопределенная процедура при установке интервала журнала.

Синтаксис:

ПриУстановкеИнтервала(<ДатаНач>, <ДатаКон>)

Англоязычный синоним:

OnSetSelectInJournal

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

ПриПоказеПроводокПоДокументу

Предопределенная процедура при показе проводок по документу (операции).

Синтаксис:

ПриПоказеПроводокПоДокументу(<Документ>)

Англоязычный синоним:

OnShowEntrysByDoc

Параметры:

<Документ>	Документ по которому выводятся проводки.

Описание:

При работе с журналом операций или журналом документов пользователь имеет возможность включить режим показа проводок текущей операции. При этом открывается окно журнала проводок «синхронизированного» с текущим журналом. В этом журнале проводок будут выдаваться для просмотра проводки операции, на которой стоит курсор в журнале документов(операций). При установке курсора на другой документ(операцию) содержимое журнала проводок будет изменяться, показывая проводки по текущей операции. В журнале операций такой журнал проводок может быть выведен в нижней части окна.

Вызов предопределенной процедуры ПриПоказеПроводокПоДокументу производится системой 1С:Предприятие неявно при обновлении содержимого подчиненного журнала проводок текущей операции.

Переданный в качестве параметра документ может быть использован для определения необходимости показха проводок по данной операции.

Если в данной предопределенной процедуре установлен статус возврата 0, то проводки операции показываться не будут (например, если пользователю запрещен просмотр проводок).

Пример:

```
Процедура ПриПоказеПроводокПоДокументу(ТекДок)
   Если ТекДок.Операция.Фирма <> ТекФирма Тогда
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```
См. также: СтатусВозврата

Глава 20
Работа с бухгалтерскими итогами

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 количственного учета, а также включение аналитического учета по субконто.

Изменение бухгалтерских итогов может производиться только проводками бухгалтерских операций.

Объект встроенного языка «БухгалтерскиеИтоги» предназначен для организации доступа к итогам в различных разрезах, за различные периоды и с разной степенью детализации.

Контекст работы с бухгалтерскими итогами

Обращение к бухгалтерским итогам выполняется посредством агрегатного объекта типа «БухгалтерскиеИтоги» (Англоязычный синоним: `BookkeepingTotals`). Объект такого типа должен быть создан при помощи функции `СоздатьОбъект`

```
БухИтоги = СоздатьОбъект("БухгалтерскиеИтоги");
```

Таких объектов может быть в системе несколько (любое необходимое количество). Объект может создаваться непосредственно перед использованием или в глобальном модуле (с объявлением экспортируемой переменной). При этом следует иметь в виду, что различные установки, назначаемые объекту будут действовать до их переустановки или удаления объекта. Поэтому обычно данный объект создается перед его использованием.

Механизм бухгалтерских итогов, поддерживаемый компонентой «Бухгалтерский учет» системы 1С:Предприятие релизует хранение накопленных итогов для обеспечения быстрого обращения к ним при составлении отчетов и выполнения различных вычислений.

Хранение итогов поддерживается системой с детализацией до месяца. Кроме того, хранятся не все возможные итоги, а те, обращение к которым выполняется наиболее часто — это остатки и обороты по счетам с детализацией по объектам аналитики (субконто), а также обороты между счетам (без учета аналитики).

Обращение к этим итогам выполняется системой непосредственно. Для получения других итогов (с детализацией меньше месяца, с получением оборотов между различными объектами аналитики, а также сложных выборок) требуется выполнение предварительных действий — временного расчета или запроса.

Объект «БухгалтерскиеИтоги» может работать в 3-х различных режимах:

- работа с основными итогами;
- работа с временными итогами;
- работа в режиме запроса;

При создании объекта он работает в режиме работы с основными итогами. Переключение его в остальные режимы выполняется методами `Рассчитать` и `ВыполнитьЗапрос`. В зависимости от режима изменяется состав и использование атрибутов и методов объекта.

Кроме того некоторые установки объекта влияют на получение итогов во всех режимах. К ним относятся установки используемого плана счетов и разделителя учета.

Общие свойства

Работа объекта «БухгалтерскиеИтоги» во всех режимах имеет некоторые общие особенности.

Значение типа «Счет» в параметры методов объекта может передаваться в виде строки, содержащей код счета. При этом счет определяется исходя из текущей установки используемого плана счетов основного плана счетов, заданного в метаданных.

Там, где методам объекта в качестве параметра должна передаваться валюта должно передаваться значение типа того справочника, который выбран в настройке планов счетов в качестве справочника валют. В тексте описания объекта такие значения будут обозначаться как значение типа «Справочник.Валюты», хотя на прикатике идентификатор справочника может быть иным, в зависимости от выбранного в настройке планов счетов значения.

При ведении учета по нескольким планам счетов или с использованием разделителя учета на получение итогов в различных режимах запроса влияют установки методов `ИспользоватьПланСчетов` и `ИспользоватьРазделительУчета`.

ИспользоватьПланСчетов

Назначить план счетов, по которому будут выдаваться итоги.

Синтаксис:

`ИспользоватьПланСчетов(<ПланСчетов>)`

Англоязычный синоним:

`UseChartOfAccounts`

Параметры:

`<ПланСчетов>`	Необязательный параметр. Значение типа «План Счетов». Если не задан установка не меняется.

Возвращаемое значение:

Значение данной установки до вызова метода.

Описание:

Применение данного метода имеет смысл только если используется несколько планов счетов.

Метод `ИспользоватьПланСчетов` задает план счетов для метода получения итогов и `ВыполнитьЗапрос` для тех случаев, когда конкретный счет не указан или задается строкой символов.

Если план счетов не установлен функцией `ИспользоватьПланСчетов`, будет использоваться основной план счетов, заданный в метаданных.

Пример:

```
БухИтоги.ИспользоватьПланСчетов(ПланыСчетов.Рабочий);
```

ИспользоватьРазделительУчета

Установить значение разделителя учета.

Синтаксис:

```
ИспользоватьРазделительУчета(<РазделительУчета>)
```

Англоязычный синоним:

```
UseAccountingDivision
```

Параметры:

`<РазделительУчета>`	Необязательный параметр. Значение разделителя учета. Если не задан, то установка не меняется.

Возвращаемое значение:

Значение данной установки до вызова метода.

Описание:

Метод `ИспользоватьРазделительУчета` задает значения разделителя учета для методов получения итогов и `ВыполнитьЗапрос`.

Применение данного метода имеет смысл только если используется разделителель учета.

Пример:

```
БухИтоги.ИспользоватьРазделительУчета(Константа.ОснФирма);
```

Работа с основными итогами

Основными итогами называются остатки и обороты по счетам и объектам аналитического учета, а также обороты между счетами за любой рассчитанный период с детализацией до месяца.

В пункте меню «Управление бухгалтерскими итогами» устанавливается последний рассчитанный период. В режиме работы с основными итогами обращение может выполняться только к итогам по рассчитанный период включительно.

В этом режиме работают два вида методов — получение остатков и оборотов, а также установка периода за который выдаются итоги.

По умолчанию используется период, выбранный пользователем интерактивно через меню «Сервис»-«Параметры»-«Бухгалтерские итоги».

Остатки и обороты по счетам

Для расчета остатков и обротов по счетам существует группа функций со сходным синтаксисом и набором параметров.

СНД, СНК, СКД, СКК, ДО, КО

`СНД`	дебетовое сальдо по счету на начало периода;
`СНК`	кредитовое сальдо по счету на начало периода;
`СКД`	дебетовое сальдо по счету на конец периода;
`СКК`	кредитовое сальдо по счету на конец периода;
`ДО`	дебетовый оборот по счету за период;
`КО`	кредитовый оборот по счету за период.

Синтаксис:

```
СНД (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...)
СНК (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...}
СКД (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...)
СКК (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...)
ДО (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...)
КО (<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>...)
```

Англоязычные синонимы:

```
IDB
ICB
FDB
FCB
TD
TC
```

Параметры:

<Счет>	Значение типа «Счет» — счет расчета итогов. Может использоваться строка — код счета.
<ТипСуммы>	Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений: 1 ("С", "S") — сумма; 2 ("В", "С") — валютная сумма; 3 ("К", "А") — количество. Если параметр не указан, метод возвращает сумму.
<Валюта>	Необязательный параметр. Значение типа «Справочник.Валюты». Если параметр не указан итоги выдаются без учета валюты.
<Субконто1> <Субконто2>…	Необязательный параметр. Значения субконто. Их количество зависит от настройки субконто для данного счета. Если параметры не указаны, итоги выдаются без учета аналитики.

Все параметры кроме счета могут не указываться.

Возвращаемое значение:

Число —сальдо или оборот.

Описание:

Функции СНД, СНК, СКД, СКК, ДО, КО возвращают суммы остатков и оборотов по указанному счету за период. Выдаваться может сумма или валютная сумма или количество, в зависимости от параметра <ТипСуммы>.

Если указана валюта, то данные выдаются по конкретной валюте. Валютная сумма может выдаваться только по конкретной валюте.

Если указаны значения субконто, то данные выдаются по конкретным объектам аналитики.

Пример:

* Вычисляются остатки на счете 51 на начало и конец периода бухгалтерских итогов. Рассчитанные остатки присваиваются переменным.

```
// Создадим объект для работы с бухгалтерскими итогами
БухИтоги = СоздатьОбъект("БухгалтерскиеИтоги);
// Вычислим остаток на расчетном счете (счет 51) на начало периода
П2623 = БухИтоги.СНД("51");
// Вычислим остаток на расчетном счете (счет 51) на конец периода
П2624 = БухИтоги.СКД("51");
```

Обороты между счетами

ОБ

Расчет оборотов между счетами.

Синтаксис:

ОБ(<СчетДеб>, <СчетКред>, <ТипСуммы>, <Валюта>)

Англоязычный синоним:

TO

Параметры:

<СчетДеб>, <СчетКред>	Значения типа «Счет» — счета дебета и кредита, для которых необходимо выдать перекрестные обороты. Может использоваться строка — код счета.
<ТипСуммы>	Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений: 1 ("С", "S") — сумма; 2 ("В", "С") — валютная сумма; 3 ("К", "А") — количество. Если параметр не указан, метод возвращает сумму.
<Валюта>	Необязательный параметр. Значение типа «Справочник.Валюты». Если параметр не указан итоги выдаются без учета валюты.

Возвращаемое значение:

Число —оборот с дебета счета <СчетДеб> в кредит счета <СчетКред>.

Описание:

Функция ОБ предназначена для выдачи перекрестных оборотов между счетами. Коды счетов, оборот между которыми требуется рассчитать, передаются функции в качестве парметров.

Если указана валюта, то данные выдаются по конкретной валюте. Валютная сумма может выдаваться только по конкретной валюте.

Пример:

```
КассаБанк = БухИтоги.ОБ("50", "51");
```

Развернутое сальдо по субсчетам

Для получения развернутого сальдо по счетам, имеющим субсчета, существует группа функций со сходным синтаксисом и набором параметров.

СНДР, СНКР, СКДР, СККР

СНДР дебетовое развернутое сальдо по субсчетам на начало периода;

СНКР кредитовое развернутое сальдо по субсчетам на начало периода;
СКДР дебетовое развернутое сальдо по субсчетам на конец периода;
СККР кредитовое развернутое сальдо по субсчетам на конец периода.

Синтаксис:

```
СНДР(<Счет>, <ТипСуммы>, <Валюта>)
СНКР(<Счет>, <ТипСуммы>, <Валюта>)
СКДР(<Счет>, <ТипСуммы>, <Валюта>)
СККР(<Счет>, <ТипСуммы>, <Валюта>}
```

Англоязычные синонимы:

```
IDBR
ICBR
FDBR
FCBR
```

Параметры:

<Счет>	Значение типа «Счет» — счет, для которого необходимо рассчитать развернутое сальдо. Может использоваться строка — код счета.
<ТипСуммы>	Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений: 1 ("С", "S") — сумма; 2 ("В", "С") — валютная сумма; 3 ("К", "А") — количество. Если параметр не указан, метод возвращает сумму.
<Валюта>	Необязательный параметр. Значение типа «Справочник. Валюты». Если параметр не указан итоги выдаются без учета валюты.

Возвращаемое значение:

Число — развернутое сальдо.

Описание:

Функции СНДР, СНКР, СКДР, СККР предназначены для расчета остатков по счетам, у которых учет ведется на субсчетах. Каждая функция из этой группы возвращает остаток как сумму соответствующих остатков (дебетовых или кредитовых) всех субсчетов указанного счета. При этом учитываются остатки по субсчетам являющихся собственно счетами, а не группами.

Если указана валюта, то данные выдаются по конкретной валюте. Валютная сумма может выдаваться только по конкретной валюте.

Пример:

* Для помещения в бухгалтерский баланс вычисляется развернутое сальдо по 68 счету, учет на котором ведется на субсчетах. Дебетовая составляющая должна попасть в актив баланса, кредитовая составляющая — в пассив.

```
// Создадим объект для работы с бухгалтерскими итогами
БухИтоги = СоздатьОбъект( "БухгалтерскиеИтоги);
// Вычислим дебетовое сальдо на 68 счете на начало периода
П2463 = БухИтоги.СНДР("68");
// Вычислим дебетовое сальдо на 68 счете на конец периода
П2464 = БухИтоги.СКДР("68");
// Вычислим кредитовое сальдо на 68 счете на начало периода
П6263 = БухИтоги.СНКР("68");
// Вычислим кредитовое сальдо на 68 счете на конец периода
П6264 = БухИтоги.СККР("68");
```

Развернутое сальдо по субконто

Для расчета развернутого сальдо по счетам, имеющим субконто, во встроенном языке существует группа функций со сходным синтаксисом и набором параметров.

СНДРС, СНКРС, СКДРС, СККРС

СНДРС дебетовое развернутое сальдо по субконто на начало периода;
СНКРС кредитовое развернутое сальдо по субконто на начало периода;
СКДРС дебетовое развернутое сальдо по субконто на конец периода;
СККРС кредитовое развернутое сальдо по субконто на конец периода.

Синтаксис:

```
СНДРС(<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>, <ТипФильтра1>, <Субконто2>,
            <ТипФильтра2>...)
СНКРС(<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>, <ТипФильтра1>, <Субконто2>,
            <ТипФильтра2>...)
СКДРС(<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>, <ТипФильтра1>, <Субконто2>,
            <ТипФильтра2>...)
СККРС(<Счет>, <ТипСуммы>, <Валюта>, <Субконто1>, <ТипФильтра1>, <Субконто2>,
            <ТипФильтра2>...}
```

Англоязычные синонимы:

IDBRS
ICBRS
FDBRS
FCBRS

Параметры:

<Счет>	Значение типа «Счет» — счет, для которого необходимо рассчитать развернутое сальдо. Может использоваться строка — код счета.
<ТипСуммы>	Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений: 1 ("C", "S") — сумма; 2 ("B", "C") — валютная сумма; 3 ("K", "A") — количество. Если параметр не указан, метод возвращает сумму.
<Валюта>	Необязательный параметр. Значение типа «Справочник. Валюты». Если параметр не указан итоги выдаются без учета валюты.
<Субконто1>	Необязательный параметр. Значение субконто 1.
<ТипФильтра1>	Необязательный параметр. Тип использования субконто 1. (Строка или число). "*" (1) — Разворачивать по данному субконто. "!" (2) — Отбирать по данному субконто. " " (3) — Не учитывать данное субконто. Значения по умолчанию: "*" (1) — для первого субконто, " " (3) — для остальных субконто.
<Субконто2>	…
<ТипФильтра2>	…

Возвращаемое значение:

Число — развернутое сальдо.

Описание:

Функции СНДРС, СНКРС, СКДРС, СККРС предназначены для расчета остатков по счетам, у которых ведется аналитический учет по субконто. Каждая функция из этой группы возвращает остаток как сумму соответствующих остатков (дебетовых или кредитовых) по всем объектам аналитического учета.

Если аналитический учет по счету ведется по двум и более субконто, то для развернутого сальдо параметр <ТипФильтра> устанавливает участие данног вида субконто в получении развернутого сальдо. В зависимости от значения этого параметра данное субконто может участвовать в «развороте» остатка, или накладывать дополнительное ограничение (отбирать) на анализируемые остатки, или никак не влиять на результат функции.

Пример:

* Получим развернутое сальдо дебетовое на начало периода по счету 60, учет ведется по субконто Организации.

```
СР60 = БухИтоги.СНДРС("60");
```

* Получим развернутое сальдо дебетовое на начало периода по счету 10 конкретному складу в разрезе материалов, учет ведется по субконто «Материалы» и «Склады».

```
СР10 = БухИтоги.СНДРС("10", 1, , , "*", ВыбСклад, "!");
```

Установка периода итогов

ПериодД

Устанавливает произвольный период в качестве периода расчета итогов.

Синтаксис:

ПериодД(<ДатаНачалаПериода>, <ДатаКонцаПериода>)

Англоязычный синоним:

PeriodD

Параметры:

<ДатаНачалаПериода>	Необязательный параметр. Начальная дата периода выдачи итогов.
<ДатаКонцаПериода>	Необязательный параметр. Конечная дата периода выдачи итогов.

Описание:

Метод ПериодД задает период расчета итогов для функций режима основных итогов.

Период, установленный методом ПериодД, действует до конца вызова другого метода установки периода для этого объекта.

Если параметры не заданы — значения периода берется из периода, заданного пользователем интерактивно в меню «Сервис»-«Параметры»-«Бухгалтерские итоги».

Пример:

```
БухИтоги.ПериодД('07.01.98', '17.01.98');
```

ПериодКВ

Устанавливает квартал в качестве периода расчета итогов.

Синтаксис:

ПериодКВ(<Дата>|<НомерКвартала>, <Год>)

Англоязычный синоним:

PeriodQ

Параметры:

<Дата>	Дата — любая дата из квартала, устанавливаемого в качестве периода расчета итогов.
<НомерКвартала>	Число от 1 до 4 — порядковый номер квартала.
<Год>	Год заданный числом (включая век). Используется только, если первый параметр — номер квартала.

Описание:

Метод ПериодКВ задает период расчета итогов для функций режима основных итогов. В качестве параметра метода может передаваться либо дата, лежащая в том квартале, который будет назачен в качестве периода рачета итогов, либо номер квартал — цифра от 1 до 4. Номер квартала задает квартал года, указанного вторым параметром.

Период, установленный методом ПериодКВ, действует до вызова другго метода установки периода.

Пример:

```
БухИтоги.ПериодКВ('07.01.98');
```

ПериодКВН

Устанавливает в качестве периода расчета итогов период с начала года до конца указанного квартала.

Синтаксис:

```
ПериодКВН(<Дата>|<НомерКвартала>, <Год>)
```

Англоязычный синоним:

```
PeriodQAccumY
```

Параметры:

<Дата>	Дата — любая дата квартала, устанавливаемого в качестве периода расчета итогов.
<НомерКвартала>	Число от 1 до 4 — порядковый номер квартала.
<Год>	Год заданный числом (включая век). Используется только, если первый параметр — номер квартала.

Описание:

Метод ПериодКВН задает период расчета итогов для функций режима основных итогов. В качестве периода расчета итогов метод устанавливает период с начала года до конца квартала, задаваемого параметрами.

В качестве параметра метода может передаваться либо дата, лежащая в том квартале, который будет назачен в качестве периода рачета итогов, либо номер квартал — цифра от 1 до 4. Номер квартала задает квартал года, указанного вторым параметром.

Периодом расчета итогов считается период с начала года до конца указанного квартала.

Период, установленный методом ПериодКВН, действует до вызова другого метода установки периода.

Пример:

```
БухИтоги.ПериодКВН('07.01.98');
```

ПериодМ

Устанавливает в качестве периода расчета итогов месяц.

Синтаксис:

```
ПериодМ(<Дата>|<НомерМесяца>, <Год>)
```

Англоязычный синоним:

```
PeriodM()
```

Параметры:

<Дата>	Дата — любая дата из месяца, устанавливаемого в качестве периода расчета итогов.
<НомерМесяца>	Число от 1 до 12 — порядковый номер месяца.
<Год>	Год заданный числом (включая век). Используется только, если первый параметр — номер месяца.

Описание:

Метод ПериодМ задает период расчета итогов для функций режима основных итогов. В качестве параметра метода может передаваться либо дата месяца, который будет назначен в качестве периода рачета итогов, либо номер месяца — число от 1 до 12. Номер месяца задет месяц года переданного в качестве второго пареметра.

Период, установленный методом ПериодМ, действует до вызова другого метода установки периода.

Пример:

```
БухИтоги.ПериодМ('07.01.98');
```

ПериодМНК

Устанавливает в качестве периода расчета итогов период с начала квартала до конца указанного месяца.

Синтаксис:

```
ПериодМНК(<Дата>|<НомерМесяца>, <Год>)
```

Англоязычный синоним:

```
PeriodMAccumQ
```

Параметры:

<Дата>	Дата — любая дата месяца, устанавливаемого в качестве периода расчета итогов.
<НомерМесяца>	Число от 1 до 12 — порядковый номер месяца.
<Год>	Год заданный числом (включая век). Используется только, если первый параметр — номер месяца.

Описание:

Метод ПериодМНК задает период расчета итогов для функций режима основных итогов. В качестве периода расчета итогов метод устанавливает период с начала квартала до конца месяца, задаваемого параметрами.

В качестве параметра метода может передаваться либо дата месяца, который будет использован в качестве периода расчета итогов, либо номер месяца — число от 1 до 12. Номер месяца задает месяц года, заданного вторым параметром.

Периодом расчета итогов считается период с начала квартала до конца указанного месяца.

Период, установленный методом ПериодМНК, действует до вызова другой функции установки периода.

Пример:

```
БухИтоги.ПериодМНК('07.08.98');
```

ПериодМНГ

Устанавливает в качестве периода расчета итогов период с начала года до конца указанного месяца.

Синтаксис:

```
ПериодМНГ(<Дата>|<НомерМесяца>, <Год>}
```

Англоязычный синоним:

```
PeriodMAccumY
```

Параметры:

<Дата>	Дата — любая дата месяца, устанавливаемого в качестве периода расчета итогов.
<НомерМесяца>	Число от 1 до 12 — порядковый номер месяца.
<Год>	Год заданный числом (включая век). Используется только, если первый параметр — номер месяца.

Описание:

Метод ПериодМНГ задает период расчета итогов для функций режима основных итогов. В качестве периода расчета итогов функция устанавливает период с начала года до конца месяца, задаваемого параметрами.

В качестве параметра метода может передаваться либо дата месяца, который будет использован в качестве периода расчета итогов, либо номер месяца — число от 1 до 12. Номер месяца задает месяц года, заданного вторым параметром.

Периодом расчета итогов считается период с начала года до конца указанного месяца.

Период, установленный функцией ПериодМНГ, действует до вызова другой функции установки периода.

Пример:

```
БухИтоги.ПериодМНГ('07.08.98');
```

НачПериода

Выдает начальную дату установленного в данный момент периода основных итогов.

Синтаксис:

```
НачПериода()
```

Англоязычный синоним:

```
BeginOfPeriod
```

Возвращаемое значение: Начальная дата периода.

Описание:

Метод НачПериода используется для определения начальной даты основных итого выдаваемых объектом.

Пример:

```
Сообщить(ПериодСтр(БухИтоги.НачПериода(), БухИтоги.КонПериода());
```

КонПериода

Выдает конечную дату установленного в данный момент периода основных итогов.

Синтаксис:

```
КонПериода()
```

Англоязычный синоним:

```
EndOfPeriod
```

Возвращаемое значение: Конечная дата периода.

Описание:

Метод КонПериода используется для определения конечной даты основных итогов выдаваемых объектом.

Пример:

```
Сообщить(ПериодСтр(БухИтоги.НачПериода(), БухИтоги.КонПериода());
```

ОсновныеИтоги

Переводит объект в режим работы с основными итогами.

Синтаксис:

```
ОсновныеИтоги()
```

Англоязычный синоним:

```
MainTotals
```

Описание:

Метод ОсновныеИтоги переводит объект в режим работы с основными итогами. Вызов этого метода имеет смысл тогда, когда был выполнен расчет временных итогов или запрос, и нужно вернуть объект к работе с основными итогами. При этом результаты запроса или расчета временных итогов теряются.

Пример:

```
БухИтоги.ОсновныеИтоги();
```

Работа с временными итогами

Для переключения в режим работы с временными итогами следует выполнить метод Рассчитать. После этого объект позволяет получать остатки и обороты по счетам за произвольный период, заданный при вызове метода Рассчитать.

Для получения временных итогов используются теже методы, что и для получения основных итогов (СНД, СНК , СКД , СКК, ДО, КО, ОБ и другие). Параметры метода Рассчитать определяют итоги, которые будут доступны для получения этими методами.

Рассчитать

Расчет временных итогов.

Синтаксис:

```
Рассчитать(<НачалоПериода>, <КонецПериода> , <ФильтрПоСчетам>, <ТолькоСинтетика> , <ПланСчетов>, <РазделительУчета>)
```

Англоязычный синоним:

```
Calculate
```

Параметры:

<НачалоПериода> Необязательный параметр. Выражение типа дата, документ или позиция документа начала периода расчета временных итогов. Если этот параметр не указан, будут вычисляться конечные сальдо на момент, указанный в параметре <КонецПериода>.

<КонецПериода> Необязательный параметр. Выражение типа дата, документ или позиция документа конца периода расчета временных итогов. Если этот параметр не указан, будут вычисляться начальные сальдо на момент, указанный в параметре <НачалоПериода>.

<ФильтрП, рСчетам> Необязательный параметр. Счета, для которых

будет выполняться временный расчет итогов. Задается значением типа «Счет» или объектом типа «СписокЗначений», содержащим значения типа «Счет», либо строкой содержащей список кодов счетов, разделенных символом ", " или ";".

<ТолькоСинтетика> Необязательный параметр:

1 — рассчитывать сальдо только по счетам;

0 — или не указан — рассчитывать сальдо по счетам и по субконто.

<ПланСчетов> Необязательный параметр. Значение типа «План Счетов». Ограничение расчета одним планом счетов.

<РазделительУчета> Необязательный параметр. Значение разделителя учета. Ограничение расчета одним значением разделителя учета.

Возвращаемое значение:

Число: 1 — расчет выполнен; 0 — расчет не выполнен.

Описание:

Метод Рассчитать выполняет расчет итогов за период, задаваемый параметрами <НачалоПериода> и <КонецПериода>. Метод задает период, за который будут возвращать остатки и обороты по счетам функции СНД, СКД, снк, скк, до, ко, ОБ и другие.

Если из двух параметров периода задан только параметр начала периода, метод Рассчитать вычислит начальные остатки — без оборотов. Наоборот, если задан параметр конца периода, будут вычисленные конечные остатки.

Параметр <ФильтрПоСчетам> позволяет рассчитать итоги только для конкретного счета или для группы счетов.

Установка параметра <ТолькоСинтетика> в 1 задает выполнение расчета итогов только по счетам, без субконто. В этом случае функции для получения развернутого сальдо по субконто СНДРС, СКДРС, СНКРС, СККРС для счетов, по которым ведется аналитический учет по субконто будут возвращать 0, даже если реально в информационной базе есть остатки по этим счетам.

Если параметр <ТолькоСинтетика> не указан или равен 0, временный расчет итогов будет выполнен и по счетам, и по субконто.

При использовании методов ВыполнитьЗапрос и Рассчитать возможно указание только одной границы интервала. В этом случае будут рассчитываться только остатки на эту границу.

Пример:

```
БухИтоги.Рассчитать(ДатаНач, ДатаКон);
```

Актуальность

Устанавливает или сбрасывает признак актуальности временных итогов.

Синтаксис:

```
Актуальность(<Флаг>)
```

Англоязычный синоним:

```
Actual
```

Параметры:

<Флаг> Необязательный параметр. Число — признак актуальности временного расчета итогов. Параметр может принимать значения:

1 — поддерживать временный расчет в актуальном состоянии;
О—не поддерживать временный расчет в актуальном состоянии;
Если параметр не задан, то значение не меняется.

Возвращаемое значение:

Значение признака актуальности на момент до вызова функции.

Описание:

Данный метод позволяет организовать поддержку временного расчета итогов в актуальном состоянии. При установке объекту «БухгалтерскиеИтоги»

признака поддержки в актуальном состоянии, в нем будут отражаться все изменения в итогах, выполняемые операциями. При этом данный актуальный объект «БухгалтерскиеИтоги» смогут автоматически использовать временные расчеты и запросы других объектов «БухгалтерскиеИтоги» с аналогичными фильтрами, что позволяет оптимизировать время расчета.

Данную возможность следует использовать только в специальных случаях, например, для оптимизации больших регламентных расчетов.

Данная возможность может применяться только при работе программы в монопольном режиме или, если не в монопольном режиме, то только при проведении документа.

Пример:

```
БухИтоги.Актуальность(1);
```

Работа в режиме запроса

Для получения большого количества итогов в различных разрезах объект «Бухгалтерские Итоги» переключается в режим работы с запросом. Для переключения в этот режим вызывается метод `ВыполнитьЗапрос` в параметрах которого, а также дополнительными методами устанавливается состав итогов, которые будут получены запросом. Выполнение данного метода осуществляет выборку данных и их предварительную обработку.

После выполнения запроса с помощью специального набора методов объекта «Бухгалтерские Итоги» осуществляется получение итогов, полученных данным запросом. К этим методам относятся методы обхода итогов в различных разрезах (сформированных запросом) и методы для получения собственно итогов. Кроме того специальный набор атрибутов объекта «Бухгалтерские Итоги» позволяет обращаться процессе обхода результатов запроса к значениям полученных группировок.

ВыполнитьЗапрос

Выборка итогов в различных разрезах.

Синтаксис:

```
ВыполнитьЗапрос(<НачалоПериода>, <КонецПериода>, <ФильтрПоСчетам>,
       <ФильтрПоКоррСчет>, <Валюта>, <ТипИтогов>, <Периодичность>, <ТипСуммы>)
```

Англоязычный синоним:

```
DoQuery
```

Параметры:

`<НачалоПериода>`	Необязательный параметр. Выражение типа дата, документ или позиция документа начала периода запроса. Если этот параметр не указан, будут вычисляться начальные сальдо на момент, указанный в параметре `<КонецПериода>`
`<КонецПериода>`	Необязательный параметр. Выражение типа дата, документ или позиция документа конца периода запроса. Если этот параметр не указан, будут вычисляться начальные сальдо на момент, указанный в параметре `<НачалоПериода>`.
`<ФильтрПоСчетам>`	Необязательный параметр. Счета, для которых будут отбираться итоги в запросе. Задается значением типа «Счет» или объектом типа «Список-Значений», содержащим значения типа «Счет», либо строкой содержащей список кодов счетов, разделенных символом ", " или ";". Если параметр не указан, отбор будет выполняться по всем счетам.
`<ФильтрПоКоррСчет>`	Необязательный параметр. Значение типа «Счет» — корреспондирующий счет, в корреспонденции с которым будут отбираться итоги счета, указанного в параметре `<ФильтрПоСчетам>`. Задается значением типа «Счет» или объектом типа «Список-Значений», содержащим значения типа «Счет», либо строкой содержащей список кодов счетов, разделенных символом "," или ";". Если параметр не указан, будут отбираться итоги в корреспонденции со всеми счетами.
`<Валюта>`	Необязательный параметр. Значение типа «Справочник.Валюты». Если параметр не указан итоги выдаются без учета валюты.
`<ТипИтогов>`	Необязательный параметр. Число — тип отбираемых итогов. Может принимать следующие значения: 1 — остатки и обороты по счету в целом; 2 — обороты между счетами; Первое и второе вместе. По умолчанию — 1.
`<Периодичность>`	Необязательный параметр. Позволяет получить дополнительный разрез итогов по периодам. Число или символьная строка (См. ниже). По умолчанию периодичность не задана.
`<ТипСуммы>`	Необязательный параметр. Число или строка — тип рассчитываемых итогов. Может

принимать следующие значения (в скобках указаны строковые синонимы):
1 («С», «S») рассчитывать суммы;
2 («В», «С») рассчитывать валютные суммы;
4 («К», «А») рассчитывать количество.
Если требуется одновременно рассчитывать разные суммы, значение параметра получается путем сложения допустимых значений, например: 5 (1+4) — рассчитывать суммы и количество. При указании параметра строкой в ней указываются все символы, которые обозначают типы сумм, которые нужно рассчитывать. По умолчанию рассчитываются все типы сумм.

Параметр <Периодичность> может принимать следующие значения (в скобках указаны строковые синонимы):

1 («Период», «Period»)	Промежуточные итоги не рассчитываются;
2 («Операция», «Entry»)	Промежуточные итоги рассчитываются по операциям;
3 («Проводка», «Operation»)	По проводкам;
4 («День», «Day»)	По дням;
5 («Неделя», «Week»)	По неделям;
6 («Декада», «Decade»)	По декадам;
7 («Месяц», «Month»)	По месяцам;
8 («Квартал», «Quorter»)	По кварталам
9 («Год», «Year»)	По годам.

Возвращаемое значение:

Число: 1 — запрос выполнен; 0 — запрос не выполнен.

Описание:

Метод ВыполнитьЗапрос выполняет отбор и расчет итогов за период, задаваемый параметрами <НачалоПериода> и <КонецПериода>.

В зависимости от переданных параметров итоги могут быть подготовлены методом ВыполнитьЗапрос в различных разрезах. Кроме того, на результат запроса влияют дополнительные установки, которые выполняются вызовами специальных методов объекта «Бухгалтерские Итоги».

Параметр <Периодичность> позволяет получить итоги запроса в разрезе периодов.

Метод ВключатьСубсчета позволяет получить итоги в разрезе субсчетов.

Применение метода ИспользоватьСубконто (перед вызовом ВыполнитьЗапрос) позволяет сформировать запрос в разрезе субконто.

После выполнения запроса обход запросов в различных разрезах выполняется соответствующими методами объекта.

Для доступа к итогам в разрезе периодов используются методы ВыбратьПериоды и ПолучитьПериод.

При использовании методов ВыполнитьЗапрос и Рассчитать возможно указание только одной границы интервала. В этом случае будут рассчитываться только остатки на эту границу.

Данный метод может использовать предварительно выполненный временный расчет. Для этого у временного расчета должен быть взведен флаг актуальности, система должна находиться в монопольном режиме, или расчет производится в модуле документа и в данном модуле выполняется запрос.

Пример:

В монопольном режиме:

```
// модуль обработки
ИтРасчет.Актуальность(1);
   ...
   Док.ВыбратьДокументы();
   Пока Док.ПолучитьДокумент() = 1 Цикл
      ИтРасчет.Рассчитать(, Док.ТекущийДокумент());
      ...
      Док.Провести();
   КонецЦикла;
   ...
// модуль документа
...
ИтЗапрос.ВыполнитьЗапрос(, Док.ТекущийДокумент(), СчетПоКоду("41"));
...
```

в данном случае ИтЗапрос.ВыполнитьЗапрос(...) воспользуется результатами временного расчета ИтРасчет. Данная возможность может использоваться при групповом перепроведении документов.

Пример:

```
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон, "60", , 3);
```

ВключатьСубсчета

Устанавливает режим отбора итогов методом ВыполнитьЗапрос по субсчетам.

Синтаксис:

```
ВключатьСубсчета(<ФлагСчета>, <ФлагКоррСчета>}
```

Англоязычный синоним:

```
IncludeSubAccounts
```

Параметры:

<ФлагСчета>	Необязательный параметр. Признак развертывания сальдо по субсчетам основного счета. Число: 0 — не разворачить по субсчетам; 1 — разворачивать по субсчетам; -1 (минус единица) — не выдавать итоги по счетам-группам. Значение по умолчанию — 0.
<ФлагКоррСчета>	Необязательный параметр. Признак развертывания сальдо по субсчетам корреспондирующего счета. Число: 0 — не разворачить по субсчетам; 1 — разворачивать по субсчетам; -1 (минус единица) — не выдавать итоги по корреспондирующим счетам-группам. Значение по умолчанию — 0.

Описание:

Метод ВключатьСубсчета должен вызываться до выполнения метода ВыполнитьЗапрос. Он устанавливает режим, при котором итоги отбираемые запросом будут разворачиваться по субсчетам соответственно основных и корреспондирующих счетов.

Если метод не вызвался, разворот итогов по субсчетам не производится.

Пример:

```
БухИтоги.ВключатьСубсчета(1, 1);
```

Опции

Устанавливает режим включения сумм в итоги.

Синтаксис:

```
Опции(<ВключатьЗабалансовыеСуммы>, <ВключатьОборотныеСубконтоСуммы>)
```

Англоязычный синоним:

```
Options
```

Параметры:

<ВключатьЗабалансовыеСуммы>	Признак включения в итоги сумм по забалансовым счетам. Число: 0 — не включаются суммы по забалансовым счетам; 1 — включаются суммы по забалансовым счетам.
<ВключатьОборотныеСубконтоСуммы>	Признак включения в итоги сумм по оборотным субконто. Число: 0 — не включаются суммы по оборотным субконто; 1 — включаются суммы по оборотным субконто.

Описание:

Метод Опции должен вызываться до выполнения метода ВыполнитьЗапрос. Он устанавливает режимы включения в итоги сумм по забалансовым счетам и оборотным субконто.

Если метод не вызвался, суммы по забалансовым счетам и оборотным субконто не включаются в запрос.

Пример:

```
БухИтоги.Опции(1, 1);
```

ИспользоватьСубконто

Устанавливает режим получения итогов методом ВыполнитьЗапрос в разрезе субконто.

Синтаксис:

```
ИспользоватьСубконто(<ВидСубконто>, <Субконто>, <ТипФильтра>, <ПоГруппам>)
```

Англоязычный синоним:

```
UseSubconto
```

Параметры:

<ВидСубконто>	Значение типа «Вид Субконто» — расчет временных итогов будет выполнен только для субконто указанного вида. Задается значением типа «Вид Субконто» или строкой содержащей имя идентификатора вида субконто, как он назван в конфигураторе.
<Субконто>	Необязательный параметр. Здесь может быть задано или одно конкретное значение субконто, по которому будут отобраны итоги по аналитике или объект типа «Список значений», где можно задать несколько значений субконто. Если параметр не задан — то считается пустым значением субконто.
<ТипФильтра>	Необязательный параметр. Число — тип фильтра по субконто. Может принимать следующие значения: 1 — разворачивать по данному субконто, 2 — отбирать по данному субконто, 3 — не учитывать это субконто вообще. По умолчанию 1.
<ПоГруппам>	Необязательный параметр. Число — группировка др итогов по субконто. Параметр <ПоГруппам> имеет смысл, если параметр <ТипФильтра> равен 1, а вид субконто, заданный параметром <ВидСубконто>, имеет тип значения «Справочник». Параметр может принимать значения: 0 — не показывать итоги по группам справочника; 1 — показывать итоги по группам справочника.

По умолчанию 0.

Описание:

Метод ИспользоватьСубконто устанавливает режим отбора итогов по субконто методом ВыполнитьЗапрос.

Метод ИспользоватьСубконто слудет вызывать до вызова метода ВыполнитьЗапрос. После выполнения метода ВыполнитьЗапрос установки метода ИспользоватьСубконто сбрасываются и перед слудеющим запросом их нужно устанавливать заново.

Метод ИспользоватьСубконто может вызываться последовательно несколько раз. В этом случае установки, выполняемые этим методом суммируются.

Использование параметра <ТипФильтра> со значением 1 (разворачивать) в сочетании с выбранной группой справочника в параметре <Субконто> позволяет получить итоги по всем элементам данной группы.

Пример:

* Данный запрос формирует итоги по счету 10 в разрезе материалов по одному складу.

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Материалы, , 1);
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Склады, ВыбСклад, 2);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон, "10");
```

ИспользоватьКорСубконто

Устанавливает режим получения итогов методом ВыполнитьЗапрос в разрезе корреспондирующих субконто.

Синтаксис:

```
ИспользоватьКорСубконто(<ВидСубконто>, <Субконто>, <ТипФильтра>, <ПоГруппам>)
```

Англоязычный синоним:

```
UseCorSubconto
```

Параметры:

<ВидСубконто>	Значение типа «Вид Субконто» — расчет временных итогов будет выполнен только для субконто указанного вида.
<Субконто>	Необязательный параметр. Значение субконто, по которому будут отобраны итоги по аналитике. Если параметр не задан — то считается пустым значением субконто.
<ТипФильтра>	Необязательный параметр. Число — тип фильтра по субконто. Может принимать следующие значения: 1 — разворачивать по данному субконто, 2 — отбирать по данному субконто, 3 — не учитывать это субконто вообще. По умолчанию 1.
<ПоГруппам>	Необязательный параметр. Число — группировка итогов по субконто. Параметр <ПоГруппам> имеет смысл, если параметр <ТипФильтра> равен 1, а вид субконто, заданный параметром <ВидСубконто>, имеет тип значения «Справочник». Параметр может принимать значения; 0 — не показывать итоги по группам справочника; 1 — показывать итоги по группам справочника. По умолчанию 0.

Описание:

Метод ИспользоватьКорСубконто устанавливает режим отбора итогов по субконто методом ВыполнитьЗапрос.

Метод ИспользоватьКорСубконто слудет вызывать до вызова метода ВыполнитьЗапрос. После выполнения метода ВыполнитьЗапрос установки метода ИспользоватьКорСубконто сбрасываются и перед слудеющим запросом их нужно устанавливать заново.

Метод ИспользоватьКорСубконто может вызываться последовательно несколько раз. В этом случае установки, выполняемые этим методом суммируются.

Использование параметра <ТипФильтра> со значением 1 (разворачивать) в сочетании с выбранной группой справочника в параметре <Субконто> позволяет получить итоги по всем элементам данной группы.

Данный метод применяется для расчета перекрестных оборотов между объектами аналитического учета.

Пример:

* Данный запрос формирует итоги по корреспонденции субконто Товары и Клиенты.

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ИспользоватьКорСубконто(ВидыСубконто.Клиенты, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
```

Работа с результатами запроса

Атрибуты и методы, описанные в этом разделе, могут использоваться для обработки результатов работы метода ВыполнитьЗапрос.

Методы выборки результатов запроса

ВыбратьСчета

Открывает выборку счетов, для которых были получены итоги методом ВыполнитьЗапрос.

Синтаксис:
ВыбратьСчета(<ФлагВсе>, <ФлагДК>, <Номер>, <РазвСальдо>)
Англоязычный синоним:
SelectAccounts
Параметры:

<ФлагВсе>	Необязательный параметр. 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все счета, которые имели итоги в данном запросе; -1, -2 ... — включить в выборку счета, которые имели итоги в группировке n-го вышестоящего уровня. По умолчанию — 0.
<ФлагДК>	Необязательный параметр. 1 — включать в выборку счета только с дебетовыми оборотами; 2 — включать в выборку счета только с кредитовыми оборотами. 0 — включать в выборку счета вне зависимости от дебетовых/кредитовых оборотов. По умолчанию 0.
<Номер>	Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке присваивается номер 0.
<РазвСальдо>	Необязательный параметр. Признак необходимости рассчитывать развернутое сальдо по субконто. Используется только если в запросе участвуют субконто. 1 — рассчитывать развернутое сальдо. 0 — не рассчитать развернутое сальдо; По умолчанию 0.

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 должен использоваться после получения итогов методом ВыполнитьЗапрос. Метод открывает выборку счетов, вошедших в запрос.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Параметр <РазвСальдо> может указываться только, если в запросе установлено участие субконто. Позволяет указать, рассчитывать ли равернутое сальдо для субконто выборки. Результаты расчета могут быть получены при помощи функций СНДРС , СНКРС , СКДРС , СККРС.

Непосредственно извлечение счетов из выборки осуществляется при помощи метода ПолучитьСчет. Счета извлекаются в порядке возрастания кода счета.

Выборке, открываемой методом ВыбратьСчета, может быть присвоен номер — положительное целое число. Номер — это «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьСчет. Использование этой метки необходимо, если требуется получить две группировки по счету.
Пример:
```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 Цикл
   Деб = БухИтоги.ДО();
   Кред = БухИтоги.КО();
КонецЦикла;
```

ПолучитьСчет

Получить из выборки следующий счет. Выборка должна быть предварительно открыта при помощи метода ВыбратьСчета.
Синтаксис:
ПолучитьСчет(<Номер>, <Счет>)
Англоязычный синоним:
GetAccount
Параметры:

<Номер>	Необязательный параметр. Положительное целое число — номер выборки.
<Счет>	Необязательный параметр. Конкретное значение счета, которое нужно получить из выборки.

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 позволяет также обращаться к результатам выборки по номеру выборки, если предварительно было открыто несколько выборок. Номер выборки передается методу в качестве параметра <Номер>.

Следует иметь в виду, что, в отличие от аналогичного параметра метода ВыбратьСчета, параметр <Номер> метода ПолучитьСчет используется только для обращения к конкретной выборке.

Метод ПолучитьСчет позволяет также обращаться к конкретному значению результа выборки. Значение счета выборки передается методу в качестве параметра <Счет>.

Если параметр <Счет> не задан, то метод ПолучитьСчет может использоваться 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:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   Деб = БухИтоги.ДО();
   Кред = БухИтоги.КО();
КонецЦикла;
```

ВыбратьКорСчета

Открывает выборку корреспондирующих счетов, для которых были получены итоги методом ВыполнитьЗапрос.

Синтаксис:

ВыбратьКорСчета(<ФлагВсе>, <ФлагДК>, <Номер>)

Англоязычный синоним:

SelectCorAccounts

Параметры:

<ФлагВсе>	Необязательный параметр. 0 — отбирать те счета, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все счета, которые имели итоги в данном запросе; -1, -2 ... — включить в выборку счета, которые имели итоги в группировке n-го вышестоящего уровня. По умолчанию — 0.
<ФлагДК>	Необязательный параметр. 1 — включать в выборку счета только с дебетовыми оборотами; 2 — включать в выборку счета только с кредитовыми оборотами. 0 — включать в выборку счета вне зависимости от дебетовых/кредитовых оборотов. По умолчанию 0.
<Номер>	Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке присваивается номер 0.

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 ВыбратьКорСчета должен использоваться после получения итогов методом ВыполнитьЗапрос. Метод открывает выборку счетов, вошедших в запрос.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Непосредственно извлечение счетов из выборки осуществляется при помощи метода ПолучитьКорСчет. Счета извлекаются в порядке возрастания кода счета.

Выборке, открываемой методом ВыбратьКорСчета, может быть присвоен номер — положительное целое число. Номер — это «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьКорСчет. Использование этой метки необходимо, если требуется получить две группировки по счету.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 Цикл
   БухИтоги.ВыбратьКорСчета();
   Пока БухИтоги.ПолучитьКорСчет() = 1 Цикл
      Деб = БухИтоги.ДО();
      Кред = БухИтоги.КО();
   КонецЦикла;
КонецЦикла;
```

ПолучитьКорСчет

Получить из выборки следующий корреспондирующий счет. Выборка должна быть предварительно открыта при помощи метода

ВыбратьКорСчета.

Синтаксис:
ПолучитьКорСчет(<Номер>, <Счет>)
Англоязычный синоним:
GetCorAccount
Параметры:

<Номер>	Необязательный параметр. Положительное целое число — номер выборки.
<Счет>	Необязательный параметр. Конкретное значение счета, которое нужно получить из выборки.

Возвращаемое значение:
Число: 1 — следующий счет выбран успешно; 0 — следующий счет не выбран (отсутствует).
Описание:
Метод ПолучитьКорСчет выбирает очередной счет из выборки, содержащей счета из плана счетов. Перед применением метода ПолучитьКорСчет выборка должна быть открыта при помощи метода ВыбратьКорСчета. Счета извлекаются в порядке возрастания кода счета.

Метод ПолучитьКорСчет позволяет также обращаться к результатам выборки по номеру выборки. Номер выборки передается методу в качестве параметра <Номер>.

Следует иметь в виду, что, в отличие от аналогичного параметра метода ВыбратьКорСчета, параметр <Номер> метода ПолучитьКорСчет используется только для обращения к конкретной выборке.

Метод ПолучитьСчет позволяет также обращаться к конкретному значению результа выборки. Значение счета выборки передается методу в качестве параметра <Счет>.

Если параметр <Счет> не задан, то метод ПолучитьКорСчет используется для организации цикла по счетам. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьКорСчет возвращает 1.

Метод возвращает 0, когда очередной счет не выбран. Это происходит, если при предыдущем применении метода был выбран последний счет выборки.
Пример:
```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьКорСчета();
   Пока БухИтоги.ПолучитьКорСчет() = 1 цикл
      Деб = БухИтоги.КорДО();
      Кред=БухИтоги.КорКО();
   КонецЦикла;
КонецЦикла;
```

ВыбратьВалюты

Открывает выборку валют.
Синтаксис:
ВыбратьВалюты(<ФлагВсе>, <ФлагДК>, <Номер> , <РазвСальдо>, <Сортировка>)
Англоязычный синоним:
Select Currencies
Параметры:

<ФлагВсе>	Необязательный параметр. 0 — отбирать те валюты, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все валюты, которые имели итоги в данном запросе; -1, -2 ... — включить в выборку валюты, которые имели итоги в группировке n-го вышестоящего уровня. По умолчанию — 0.
<ФлагДК>	Необязательный параметр. 1 — включать в выборку валюты только с дебетовыми оборотами; 2 — включать в выборку валюты только с кредитовыми оборотами. 0 — включать в выборку валюты вне зависимости от дебетовых/кредитовых оборотов. По умолчанию 0.
<Номер>	Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке присваивается номер 0.
<РазвСальдо>	Необязательный параметр. Признак необходимости рассчитывать развернутое сальдо по субконто. Используется только если в запросе участвуют субконто. 1 — рассчитывать развернутое сальдо. 0 — не рассчитать развернутое сальдо; По умолчанию 0.
<Сортировка>	Необязательный параметр. Строка — идентификатор реквизита справочника валют, который будет использован для упорядочивания обхода валют методом ПолучитьВалюту. Если значение пустое — используется представление справочника.

Возвращаемое значение:
Число: 1 — если действие выполнено и в выборке есть хотя бы одна валюта;
0 — если действие не выполнено или в выборке нет ни одной валюты.
Описание:

Метод ВыбратьВалюты должен использоваться после выполнения метода ВыполнитьЗапрос. Метод открывает выборку валют.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Параметр <РазвСальдо> может указываться только, если в запросе установлено участие субконто. Позволяет указать, рассчитывать ли развернутое сальдо для субконто выборки. Результаты расчета могут быть получены при помощи функций СНДРС , СНКРС , СКДРС , СККРС.

Непосредственно извлечение валют из выборки осуществляется при помощи метода ПолучитьВалюту.

Выборке, открываемой методом ВыбратьВалюты, может быть присвоен номер — «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьВалюту. Использование этой метки необходимо, если требуется получить две группировки по валюте.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьВалюты();
   Пока БухИтоги.ПолучитьВалюту() = 1 цикл
      ДебВ=БухИтоги.ДО(2);
      КредВ=БухИтоги.КО(2);
   КонецЦикла;
КонецЦикла;
```

ПолучитьВалюту

Получить из выборки следующую валюту. Выборка должна быть предварительно открыта при помощи метода ВыбратьВалюты.

Синтаксис:

ПолучитьВалюту(<Номер>, <Валюта>}

Англоязычный синоним:

GetCurrency

Параметры:

<Номер>	Необязательный параметр. Положительное целое число — номер выборки.
<Валюта>	Необязательный параметр. Конкретное значение валюты, которое нужно получить из выборки.

Возвращаемое значение:

Число: 1 — следующая валюта выбрана успешно; 0 — следующий валюта не выбрана (отсутствует).

Описание:

Метод ПолучитьВалюту выбирает очередную валюту из выборки, открытой при помощи метода ВыбратьВалюты.

Метод ПолучитьВалюту позволяет также обращаться к результатам конкретной выборки по ее номеру. Номер выборки передается методу в качестве параметра <Номер>.

Метод ПолучитьВалюту позволяет также обращаться к конкретному значению результа выборки. Значение валюты выборки передается методу в качестве параметра <Валюта >.

Если параметр <Валюта> не задан, то метод ПолучитьВалюту используется для организации цикла по валютам. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьВалюту возвращает 1.

Метод возвращает 0, когда очередная валюта не выбрана. Это происходит, если при предыдущем применении метода была выбрана последняя валюта выборки.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьВалюты();
   Пока БухИтоги.ПолучитьВалюту() = 1 цикл
      ДебВ = БухИтоги.ДО(2);
      КредВ = БухИтоги.КО(2);
   КонецЦикла;
КонецЦикла;
```

ВыбратьПериоды

Открывает выборку периодов.

Синтаксис:

ВыбратьПериоды(<ФлагВсе>, <ФлагДК>, <Номер>, <РазвСальдо>)

Англоязычный синоним:

SelectPeriods

Параметры:

<ФлагВсе>	Необязательный параметр. 0 — отбирать те периоды, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все периоды, которые имели итоги в данном запросе;

-1, -2 ... — включить в выборку периоды, которые имели итоги в группировке n-го вышестоящего уровня.
По умолчанию — 0.

<ФлагДК> Необязательный параметр.
1 — включать в выборку периоды только с дебетовыми оборотами;
2 — включать в выборку периоды только с кредитовыми оборотами.
0 — включать в выборку периоды вне зависимости от дебетовых/кредитовых оборотов.
По умолчанию 0.

<Номер> Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке приваивается номер 0.

<РазвСальдо> Необязательный параметр. Признак необходимости рассчитывать развернутое сальдо по субконто. Используется только если в запросе участвуют субконто.
1 — рассчитывать развернутое сальдо;
0 — не рассчитывать развернутое сальдо.
По умолчанию 0.

Возвращаемое значение:

Число: 1 — если действие выполнено и в выборке есть хотя бы один период;
0 — если действие не выполнено или в выборке нет ни одного периода.

Описание:

Метод ВыбратьПериоды должен использоваться после вызова метода ВыполнитьЗапрос. Метод открывает выборку, содержащую периоды итогов. Получение итогов в разрезе периодов доступно только если в запросе был задан параметр, устанавливающий разворачивание итогов с некоторой периодичностью.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Параметр <РазвСальдо> может указываться только, если в запросе установлено участие субконто. Позволяет указать, рассчитывать ли равернутое сальдо для субконто выборки. Результаты расчета могут быть получены при помощи функций СНДРС , СНКРС , СКДРС , СККРС.

Непосредственно извлечение периодов из выборки осуществляется при помощи метода ПолучитьПериод.

Выборке, открываемой методом ВыбратьПериоды, может быть присвоен номер — «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьПериод. Использование этой метки необходимо, если требуется получить две группировки по периоду. Это позволяет получить две группировки но с одной периодичностью.

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, , , , , "Месяц");
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьПериоды();
   Пока БухИтоги.ПолучитьПериод() = 1 цикл
      Деб = БухИтоги.ДО();
      Кред = БухИтоги.КО();
   КонецЦикла;
КонецЦикла;
```

ПолучитьПериод

Получить из выборки следующий период. Выборка должна быть предварительно открыта при помощи метода ВыбратьПериоды.

Синтаксис:

ПолучитьПериод(<Номер>, <ДатаНачалаПериода>}

Англоязычный синоним:

GetPeriod

Параметры:

<Номер> Необязательный параметр. Положительное целое число — номер выборки.

<ДатаНачалаПериода> Необязательный параметр. Конкретное значение периода, который нужно получить из выборки.

Возвращаемое значение:

Число: 1 — следующий период выбран успешно; 0 — следующий период не выбран (отсутствует).

Описание:

Метод ПолучитьПериод выбирает очередной период из выборки, открытой при помощи метода ВыбратьПериоды.

Метод ПолучитьПериод позволяет также обращаться к результатам конкретной выборки по ее номеру. Номер выборки передается методу в качестве параметра <Номер>.

Метод ПолучитьПериод позволяет также обращаться к конкретному значению результа выборки. Значение периода выборки передается методу в качестве параметра <ДатаНачалаПериода>.

Если параметр <ДатаНачалаПериода> не задан, то метод ПолучитьПериод используется для организации цикла по периодам. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьПериод возвращает 1.

Метод возвращает 0, когда очередной период не выбран. Это происходит, если при предыдущем применении метода был выбран последний период выборки.

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, , , , , "Месяц");
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьПериоды();
   Пока БухИтоги.ПолучитьПериод() = 1 цикл
      Деб = БухИтоги.ДО();
      Кред = БухИтоги.КО();
   КонецЦикла;
КонецЦикла;
```

ВыбратьСубконто

Открывает выборку по субконто.

Синтаксис:

```
ВыбратьСубконто(<Индекс>, <ФлагВсе>, <ФлагДК>, <Номер>, <РазвСальдо>,
                        <Сортировка>, <РежимОбхода>)
```

Англоязычный синоним:

```
SelectSubconto
```

Параметры:

<Индекс>	Необязательный параметр. Число — порядковый номер вызова метода ИспользоватьСубконто.
<ФлагВсе>	Необязательный параметр. 0 — отбирать те субконто, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все субконто, которые имели итоги в данном запросе; -1, -2 ... — включить в выборку субконто, которые имели итоги в группировке n-го вышестоящего уровня. По умолчанию — 0.
<ФлагДК>	Необязательный параметр. 1 — включать в выборку субконто только с дебетовыми оборотами; 2 — включать в выборку субконто только с кредитовыми оборотами. 0 — включать в выборку субконто вне зависимости от дебетовых/кредитовых оборотов. По умолчанию 0.
<Номер>	Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке приваивается номер 0.
<РазвСальдо>	Необязательный параметр. Признак необходимости рассчитывать развернутое сальдо. 0 —не рассчитать развернутое сальдо; 1 — рассчитывать развернутое сальдо. По умолчанию 0.
<Сортировка>	Необязательный параметр. Строка — идентификатор реквизита субконто (если субконто — справочник или документ или счет), который будет использован для упорядочивания обхода субконто методом ПолучитьСубконто. Если значение пустое — используется стандартное представление.
<РежимОбхода>	Необязательный параметр. Положительное целое число — порядок выборки результатов: 0 — выборка в прямом порядке, 1 — выборка в обратном порядке; По умолчанию 0.

Возвращаемое значение:

Число: 1 — если действие выполнено и в выборке есть хотя бы одно субконто;
0 — если действие не выполнено или в выборке нет ни одного субконто.

Описание:

Метод ВыбратьСубконто должен использоваться после вызова метода ВыполнитьЗапрос. Метод открывает выборку по субконто.

Если обработка по субконто задавалась путем нескольких вызовов метода ИспользоватьСубконто, то параметр <Индекс> позволяет сослаться на субконто конкретного вида в порядке вызова метода ИспользоватьСубконто. Для этого параметром <Индекс> задается номер вызова метода ИспользоватьСубконто.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Параметр <РазвСальдо> позволяет указать, рассчитывать ли равернутое сальдо для субконто выборки.

Извлечение субконто из выборки осуществляется при помощи метода ПолучитьСубконто.

Выборке, открываемой методом ВыбратьСубконто, может быть присвоен номер — «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьСубконто. Это позволяет получить две группировки по субконто данного вида.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Материалы, , 1);
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Склады, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон, "10");
```

```
БухИтоги.ВыбратьСубконто(1);
Пока БухИтоги.ПолучитьСубконто(1) = 1 цикл
   БухИтоги.ВыбратьСубконто(2);
   Пока БухИтоги.ПолучитьСубконто(2) = 1 цикл
      Деб = БухИтоги.ДО();
      Кред = БухИтоги.КО();
   КонецЦикла;
КонецЦикла;
```

ПолучитьСубконто

Получить из выборки следующее субконто. Выборка должна быть предварительно открыта при помощи метода ВыбратьСубконто.

Синтаксис:

ПолучитьСубконто(<Индекс>, <Номер>, <Субконто>)

Англоязычный синоним:

GetSubconto

Параметры:

<Индекс>	Необязательный параметр. Число —порядковый номер вызова метода ИспользоватьСубконто.
<Номер>	Необязательный параметр. Положительное целое число — номер выборки.
<Субконто>	Необязательный параметр. Конкретное значение субконто, который нужно получить из выборки.

Возвращаемое значение:

Число: 1 — следующее субконто выбрано успешно; 0 — следующе субконто не выбрано (отсутствует).

Описание:

Метод ПолучитьСубконто выбирает очередное субконто из выборки, открытой при помощи метода ВыбратьСубконто.

Метод ПолучитьСубконто позволяет также обращаться к результатам выборки по номеру выборки. Номер выборки передается методу в качестве параметра <Номер>.

Метод ПолучитьСубконто позволяет также обращаться к конкретному значению результа выборки. Значение субконто выборки передается методу в качестве параметра <Субконто>.

Если параметр <Субконто> не задан, то метод ПолучитьСубконто используется для организации цикла по субконто. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьСубконто возвращает 1.

Метод возвращает 0, когда очередное субконто не выбрано. Это происходит, если при предыдущем применении метода было выбрано последнее субконто выборки.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Материалы, , 1);
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Склады, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон, "10");
БухИтоги.ВыбратьСубконто(1);
Пока БухИтоги.ПолучитьСубконто(1) = 1 цикл
   БухИтоги.ВыбратьСубконто(2);
   Пока БухИтоги.ПолучитьСубконто(2) = 1 цикл
      Деб = БухИтоги.ДО();
      Кред = БухИтоги.КО();
   КонецЦикла;
КонецЦикла;
```

ВыбратьКорСубконто

Открывает выборку по корреспондирующим субконто.

Синтаксис:

```
ВыбратьКорСубконто(<Индекс>, <ФлагВсе>, <ФлагДК>, <Номер>, <Сортировка>,
                   <РежимОбхода>)
```

Англоязычный синоним:

SelectCorSubconto

Параметры:

<Индекс>	Необязательный параметр. Число —порядковый номер вызова метода ИспользоватьКорСубконто.
<ФлагВсе>	Необязательный параметр. 0 — отбирать те субконто, которые имели итоги на этом уровне обхода итогов запроса; 1 — включить в выборку все субконто, которые имели итоги в данном запросе; -1, -2 ... — включить в выборку субконто, которые имели итоги в группировке п-го вышестоящего уровня. По умолчанию — 0.
<ФлагДК>	Необязательный параметр. 1 — включать в выборку субконто только с дебетовыми оборотами;

2 — включать в выборку субконто только с кредитовыми оборотами.
0 — включать в выборку субконто вне зависимости от дебетовых/кредитовых оборотов.
По умолчанию 0.

<Номер> Необязательный параметр. Положительное целое число — номер выборки. Если параметр не указан, выборке присваивается номер 0.

<Сортировка> Необязательный параметр. Строка — идентификатор реквизита субконто (если субконто — справочник или документ или счет), который будет использован для упорядочивания обхода субконто методом ПолучитьКорСубконто. Если значение пустое — используется стандартное представление.

<РежимОбхода> Необязательный параметр. Положительное целое число — порядок выборки результатов:
0 — выборка в прямом порядке,
1 — выборка в обратном порядке.
По умолчанию 0.

Возвращаемое значение:

Число: 1 — если действие выполнено и в выборке есть хотя бы одно субконто; 0 — если действие не выполнено или в выборке нет ни одного субконто.

Описание:

Метод ВыбратьКорСубконто должен использоваться после вызова метода ВыполнитьЗапрос. Метод открывает выборку по корреспондирующим субконто.

Если обработка по корреспондирующим субконто задавалась путем нескольких вызовов метода ИспользоватьКорСубконто, то параметр <Индекс> позволяет сослаться на субконто конкретного вида в порядке вызова метода ИспользоватьКорСубконто. Для этого параметром <Индекс> задается номер вызова метода ИспользоватьКорСубконто.

Дополнительные условия на содержание выборки могут накладываться параметрами <ФлагВсе> и <ФлагДК>.

Извлечение субконто из выборки осуществляется при помощи метода ПолучитьКорСубконто.

Выборке, открываемой методом ВыбратьКорСубконто, может быть присвоен номер — «метка» выборки. Номер может использоваться для обращения к результатам выборки при помощи метода ПолучитьКорСубконто. Это позволяет получить две группировки по субконто данного вида.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ИспользоватьКорСубконто(ВидыСубконто.Клиенты, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   БухИтоги.ВыбратьКорСубконто();
   Пока БухИтоги.ПолучитьКорСубконто() =1 цикл
      Деб = БухИтоги.КорДО();
      Кред = БухИтоги.КорКО();
   КонецЦикла;
КонецЦикла;
```

ПолучитьКорСубконто

Получить из выборки следующее субконто. Выборка должна быть предварительно открыта при помощи метода ВыбратьКорСубконто.

Синтаксис:

ПолучитьКорСубконто(<Индекс>, <Номер>, <КорСубконто>)

Англоязычный синоним:

GetCorSubconto

Параметры:

<Индекс> Необязательный параметр. Число —порядковый номер вызова метода ИспользоватьКорСубконто.

<Номер> Необязательный параметр. Положительное целое число — номер выборки.

<КорСубконто> Необязательный параметр. Конкретное значение корсубконто, который нужно получить из выборки.

Возвращаемое значение:

Число: 1 — следующее субконто выбрано успешно; 0 — следующе субконто не выбрано (отсутствует).

Описание:

Метод ПолучитьКорСубконто выбирает очередное корреспондирующее субконто из выборки, открытой при помощи метода ВыбратьКорСубконто.

Метод ПолучитьКорСубконто позволяет также обращаться к результатам выборки по номеру выборки. Номер выборки передается методу в качестве параметра <Номер>.

Метод ПолучитьКорСубконто позволяет также обращаться к конкретному значению результа выборки. Значение корсубконто выборки передается методу в качестве параметра <КорСубконто>.

Если параметр <КорСубконто> не задан, то метод ПолучитьКорСубконто используется для организации цикла по корреспогди-рующим субконто. Условием цикла может служить равенство 1 возвращаемого значения метода: цикл выполняется, пока метод ПолучитьКорСубконто возвращает 1.

Метод возвращает 0, когда очередное корреспондирующее субконто не выбрано. Это происходит, если при предыдущем применении метода было выбрано последнее субконто выборки.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ИспользоватьКорСубконто(ВидыСубконто.Клиенты, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   БухИтоги.ВыбратьКорСубконто();
   Пока БухИтоги.ПолучитьКорСубконто() = 1 цикл
      Деб = БухИтоги.КорДО();
      Кред = БухИтоги.КорКО();
   КонецЦикла;
КонецЦикла;
```

Атрибуты для обращения к результатам запроса

Счет

Текущий счет результатов запроса.

Синтаксис:

```
Счет
```

Англоязычный синоним:

```
Account
```

Описание:

`Счет` является атрибутом типа «Счет». При обращении к итогам запроса при помощи функций `ВыбратьСчета`, `ПолучитьСчет`, атрибут `Счет` равен текущему счету, выбранному функцией `ПолучитьСчет`.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   Сообщить(БухИтоги.Счет.Код + " ДО = " + БухИтоги.ДО());
КонецЦикла;
```

КорСчет

Корреспондирующий счет, для которого извлекаются результаты запроса.

Синтаксис:

```
КорСчет
```

Англоязычный синоним:

```
CorAccount
```

Описание:

`КорСчет` является атрибутом типа «Счет». При обращении к итогам запроса при помощи функций `ВыбратьКорСчета`, `ПолучитьКорСчет`, атрибут `Счет` равен текущему счету, выбранному функцией `ПолучитьКорСчет`.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьКорСчета();
   Пока БухИтоги.ПолучитьКорСчет() = 1 цикл
      Сообщить(БухИтоги.КорСчет.Код + " ДО = " + БухИтоги.КорДО());
   КонецЦикла;
КонецЦикла;
```

Валюта

Валюта, по которой извлекаются результаты запроса.

Синтаксис:

```
Валюта
```

Англоязычный синоним:

```
Currency
```

Описание:

Атрибут `Валюта` имеет значение «Справочник.Валюты». Он позволяет получить, текущую валюту при использовании методов `ВыбратьВалюты` и `ПолучитьВалюту`.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьВалюты();
   Пока БухИтоги.ПолучитьВалюту() = 1 цикл
```

```
      Сообщить(БухИтоги.Счет.Код + "/" + БухИтоги.Валюта + " ДО == " +
               БухИтоги.ДО());
   КонецЦикла;
КонецЦикла;
```

НачДата

Начальная дата периода, за который извлекаются результаты запроса.

Синтаксис:

```
НачДата
```

Англоязычный синоним:

```
BegDate
```

Описание:

Атрибут `НачДата` содержит начальную дату периода, за который извлекаются итоги при использовании методов обхода периодов `ВыбратьПериоды` и `ПолучитьПериод`.

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, , , , , "Месяц");
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьПериоды();
   Пока БухИтоги.ПолучитьПериод() = 1 цикл
      Сообщить("С " + БухИтоги.НачДата.Код + " по " + БухИтоги.КонДата +
               " ДО = " + БухИтоги.ДО());
   КонецЦикла;
КонецЦикла;
```

КонДата

Конечная дата периода, за который извлекаются результаты запроса.

Синтаксис:

```
КонДата
```

Англоязычный синоним:

```
EndDate
```

Описание:

Атрибут `КонДата` содержит конечную дату периода, за который извлекаются итоги при использовании методов обхода периодов `ВыбратьПериоды` и `ПолучитьПериод`.

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, , , , , "Месяц");
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьПериоды();
   Пока БухИтоги.ПопучитьПериод() = 1 цикл
      Сообщить("С " + БухИтоги.НачДата.Код + " по " + БухИтоги.КонДата +
           " ДО = " + БухИтоги.ДО());
   КонецЦикла;
КонецЦикла;
```

Операция

Текущая операция, по которой выдаются итоги при использовании запроса.

Синтаксис:

```
Операция
```

Англоязычный синоним:

```
Operation
```

Описание:

Атрибут позволяет обращаться к текущей операции, если запрос получен с детализацией итогов по периоду «Операция» и «Проводка».

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, , , , , "Проводки");
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьПериоды();
   Пока БухИтоги.ПолучитьПериод() = 1 цикл
      Сообщить("Операция " + БухИтоги.Операция.ПредставлениеПроводки());
   КонецЦикла;
КонецЦикла;
```

Методы обращения к результатам запроса

Субконто

Субконто, соответствующее текущему итогу.

Синтаксис:

```
Субконто(<Номер> | <ВидСубконто>)
```

Англоязычный синоним:

```
Subconto
```

Параметры:

`<Номер>`	Необязательный параметр. Число — номер выборки субконто.
`<ВидСубконто>`	Необязательный параметр. Значение типа «Вид субконто».

Возвращаемое значение:

Значение субконто, соответствующее текущему итогу.

Описание:

Метод `Субконто` служит для получения значения субконто, соответствующего текущему итогу.

Метод применяется для получения текущего сукбонто при использовании методов `ВыбратьСубконто` и `ПолучитьСубконто`.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   Сообщить(БухИтоги.Субконто());
КонецЦикла;
```

КорСубконто

Корреспондирующее субконто, соответствующее текущему итогу.

Синтаксис:

```
КорСубконто(<Номер>|<ВидСубконто>)
```

Англоязычный синоним:

```
CorSubconto
```

Параметры:

`<Номер>`	Необязательный параметр. Число — номер выборки корреспондирующего субконто.
`<ВидСубконто>`	Необязательный параметр. Значение типа «ВидСубконто».

Возвращаемое значение:

Значение корреспондирующего субконто, соответствующее текущему итогу.

Описание:

Метод `КорСубконто` служит для получения значения корреспондирующего субконто, соответствующего текущему итогу. Метод применяется для получения текущего сукбонто при использовании методов `ВыбратьКорСубконто` и `ПолучитьКорСубконто`.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ИспользоватьКорСубконто(ВидыСубконто.Клиенты, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   БухМтоги.ВыбратьКорСубконто();
   Пока БухИтоги.ПолучитьКорСубконто() = 1 цикл
      Сообщить(БухИтоги.Субконто() + "-" + БухИтоги.КорСубконто());
   КонецЦикла;
КонецЦикла;
```

ПредставлениеСубконто

Представление субконто, соответствующего текущему итогу.

Синтаксис:

```
ПредставлениеСубконто(<Номер>|<ВидСубконто>, <Краткое>]
```

Англоязычный синоним:

```
SubcontoPresentation
```

Параметры:

`<Номер>`	Необязательный параметр. Число — номер выборки субконто.
`<ВидСубконто>`	Значение типа «Вид субконто».
`<Краткое>`	Необязательный параметр. 0 — полное представление субконто; 1 — краткое представление субконто. По умолчанию — 0.

Возвращаемое значение:

Строковое представление для субконто, соответствующего текущему итогу.

Описание:

Представлением называется символьная строка, содержащая информацию из различных реквизитов объекта, являющегося значением субконто. Эта символьная строка может быть использована для отображения значений субконто в различных отчетах.

Представление может быть задано только для видов субконто типа «Справочник» или «Документ». Формат представления определяется в конфигураторе при редактировании свойств вида субконто.

Метод ПредставлениеСубконто позволяет получить представление для субконто, соответствующего текущему итогу.

Метод применяется для текущего сукбонто при использовании методов ВыбратьСубконто и ПолучитьСубконто.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   Сообщить(БухИтоги.ПредставлениеСубконто());
КонецЦикла;
```

ПредставлениеКорСубконто

Представление корреспондирующего субконто, соответствующего текущему итогу.

Синтаксис:

```
ПредставлениеКорСубконто(<Номер>|<ВидСубконто>, <Краткое>)
```

Англоязычный синоним:

```
CorSubcontoPresentation
```

Параметры:

<Номер>	Необязательный параметр. Число — номер выборки корреспондирующего субконто.
<ВидСубконто>	Значение типа «Вид субконто».
<Краткое>	Необязательный параметр. 0 — полное представление субконто; 1 — краткое представление субконто. По умолчанию — 0.

Возвращаемое значение:

Строковое представление для корреспондирующего субконто, соответствующего текущему итогу.

Описание:

Представлением называется символьная строка, содержащая информацию из различных реквизитов объекта, являющегося значением субконто. Эта символьная строка может быть использована для отображения значений субконто в различных отчетах.

Представление может быть задано только для видов субконто типа «Справочник» или «Документ». Формат представления определяется в конфигураторе при редактировании свойств вида субконто.

Метод ПредставлениеКорСубконто позволяет получить представление для корреспондирующего субконто, соответствующего текущему итогу.

Метод применяется для текущего корреспондирующего сукбонто при использовании методов ВыбратьКорСубконто и ПолучитьКорСубконто.

Пример:

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Товары, , 1);
БухИтоги.ИспользоватьКорСубконто(ВидыСубконто.Клиенты, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто();
Пока БухИтоги.ПолучитьСубконто() = 1 цикл
   БухИтоги.ВыбратьКорСубконто();
   Пока БухИтоги.ПолучитьКорСубконто() = 1 цикл
      Сообщить(БухИтоги.ПредставлениеСубконто() + "-" +
                  БухИтоги.ПредставлениеКорСубконто());
   КонецЦикла;
КонецЦикла;
```

ЭтоГруппа

Определяет, является ли текущее значение группироки запроса по субконто группой или нет. Имеет смысл только для субконто типа «справочник».

Синтаксис:

```
ЭтоГруппа()
```

Англоязычный синоним:

```
IsGroup
```

Возвращаемое значение:

Число: 1 — значение группировки по субконто является группой; 0 — значение группировки по субконто не является группой.

Описание:

Данный метод позволяет определить, является ли текущее значение группироки запроса по субконто группой или нет. Метод можно применять независимо от типа данных вида субконто, однако он имеет смысл только для субконто типа «справочник», а для остальных выдает 0.

Пример:

```
Пока БухИтоги.ПолучитьСубконто() = 1 Цикл
   Если БухИтоги.ЭтоГруппа() = 1 Тогда
      Таб.ВывестиСекцию("Группа");
   Иначе
      Таб.ВывестиСекцию("Строка");
   КонецЕсли;
КонецЦикла;
```

СНД, СНК, СКД, СКК, ДО, КО

СНД — дебетовое сальдо на начало периода;
СКД — дебетовое сальдо на конец периода;
СНК — кредитовое сальдо на начало периода;
СКК — кредитовое сальдо на конец периода;
ДО — дебетовый оборот за период;
КО — кредитовый оборот за период.

Синтаксис:

```
СНД(<ТипСуммы>)
СНК(<ТипСуммы>)
СКД(<ТипСуммы>}
СКК(<ТипСуммы>)
ДО(<ТипСуммы>)
КО(<ТипСуммы>)
```

Англоязычные синонимы:

```
IDB
ICB
FDB
FCB
TD
TC
```

Параметры:

<ТипСуммы>	Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений: 1 ("С", "S") — сумма; 2 ("В", "C") — валютная сумма; 3 ("К", "A") — количество. Если параметр не указан, метод возвращает сумму.

Возвращаемое значение:

Число — сальдо или оборот.

Описание:

При работе с результатами запроса, соответствующие сальдо и обороты в соответствии с текущим уровнем вложенности группировок.

Пример:

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   ОстНачД = БухИтоги.СНД();
   ОстНачК = БухИтоги.СНК();
   ОборотД = БухИтоги.ДО();
   ОборотК = БухИтоги.КО();
   ОстКонД = БухИтоги.СКД();
   ОстКонК = БухИтоги.СКК();
КонецЦикла;
```

СНДРС, СНКРС, СКДРС, СККРС

СНДРС — дебетовое развернутое сальдо на начало периода;
СНКРС — кредитовое развернутое сальдо на начало периода;
СКДРС —дебетовое развернутое сальдо на конец периода;
СККРС —кредитовое развернутое сальдо на конец периода.

Синтаксис:

```
СНДРС(<ТипСуммы>)
СНКРС(<ТипСуммы>)
СКДРС(<ТипСуммы>}
СККРС(<ТипСуммы>)
```

Англоязычный синоним:

```
IDBRS
ICBRS
```

FDBRS
FCBRS

Параметры:

<ТипСуммы> Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений:
1 ("С", "S") — сумма;
2 ("В", "С") — валютная сумма;
3 ("К", "А") — количество.
Если параметр не указан, метод возвращает сумму.

Возвращаемое значение:

Число — рассчитанное сальдо.

Описание:

При работе с итогами, полученными методом ВыполнитьЗапрос функции СНДРС, СНКРС, СКДРС, СККРС позволяют получить развернутое сальдо в контексте текущих группировок запроса.

Пример:

* В данном примере по каждой организации определяется развернутое сальдо по счетам, к которым относится вид субконто "Организации".

```
БухИтоги.ИспользоватьСубконто(ВидыСубконто.Организации, , 1);
БухИтоги.ВыполнитьЗапрос(ДатаНач, ДатаКон);
БухИтоги.ВыбратьСубконто(1);
Пока БухИтоги.ПолучитьСубконто(1) = 1 цикл
   ОстНачД = БухИтоги.СНДРС();
   ОстНачК = БухИтоги.СНКРС();
   ОстКонД = БухИтоги.СКДРС();
   ОстКонК = БухИтоги.СККРС();
КонецЦикла;
```

КорДО, КорКО

КорДО — дебетовый оборот между корреспонденциями;
КорКО — кредитовый оборот между корреспонденциями.

Синтаксис:

КорДО(<ТипСуммы>)
КорКО(<ТипСуммы>)

Англоязычные синонимы:

CorTD
CorTC

Параметры:

<ТипСуммы> Необязательный параметр. Число или строка — тип возвращаемой суммы. Может принимать одно из следующих значений:
1 ("С", "S") — сумма;
2 ("В", "С") — валютная сумма;
3 ("К", "А") — количество.
Если параметр не указан, метод возвращает сумму.

Возвращаемое значение:

Число — вычисленный оборот.

Описание:

Функции доступны только при работе с итогам, полученными методом
ВыполнитьЗапрос.

Функции КорДО и КорКО предназначены для получения корреспондирующих оборотов между счетам или между субконто. Для их использования должны быть организованы выборки методами ВыбратьСчета, ПолучитьСчет, ВыбратьКорСчета, ПолучитьКорСчет или ВыбратьСубконто, ПолучитьСубконто, ВыбратьКорСубконто, ПолучитьКорСубконто.

Пример:

* В данном примере в переменной в переменных ОбДТ и ОбКТ образуются обороты с дебета основного счета в кредит корреспондирующего и наоборот соответственно.

```
БухИтоги.Запрос(Дата1, Дата2);
БухИтоги.ВыбратьСчета();
Пока БухИтоги.ПолучитьСчет() = 1 цикл
   БухИтоги.ВыбратьКорСчета();
   Пока БухИтоги.ПолучитьКорСчет() = 1 цикл
      ОбДТ = БухИтоги.КорДО();
      ОбКТ = БухИтоги.КорКО();
   КонецЦикла;
КонецЦикла;
```

ВыбранаПоДт, ВыбранаПоКт

`ВыбранаПоДт` — определение, выбран ли итог по дебету;

`ВыбранаПоКт` — определение, выбран ли итог по кредиту.

Синтаксис:

```
ВыбранаПоДт()
ВыбранаПоКт()
```

Англоязычный синоним:

```
SelectedByDt
SelectedByKt
```

Возвращаемое значение:

Число: 0 —данный итог по дебету(кредиту) не выбран; 1 —данный итог по дебету(кредиту) выбран.

Описание:

Данные функция используются при обходе результатов запроса с детализацией периода по операциям и проводкам. Они показывают для текущей операции (проводки) попала ли она в запрос по дебетовой (кредитовой) корреспонденции.

Пример:

```
БухИтоги.ВыполнитьЗапрос(Дата1, Дата2, "50", , , , "Проводки");
БухИтоги.ВыбратьСчета();
БухИтоги.ВыбратьПериоды();
КолПрих=0;
КолРасх=0;
Пока БухИтоги.ПолучитьПериод() = 1 Цикл
   Если БухИтоги.ВыбранаПоДт() = 1 Тогда
      КолПрих = КолПрих + 1;
   КонецЕсли;
   Если БухИтоги.ВыбранаПоКт() = 1 Тогда
      КолРасх = КолРасх + 1;
   КонецЕсли;
КонецЦикла;
```

Глава 21
Работа с Корректными проводками

Для работы с корректными проводками в системе используется специальный тип данных «КорректныеПроводки».

Контекст работы с объектом «КорректныеПроводки»

У объекта типа «КорректныеПроводки» есть набор атрибутов и методов для работы с данным объектом. Во всех программных модулях доступ к атрибутам и вызов методов корректных проводок может выполняться только при помощи переменной со ссылкой на объект типа «КорректныеПроводки». Объект создается функцией `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

Для создания объекта типа «КорректныеПроводки» в качестве параметра функции `СоздатьОбъект` передается ключевое слово «КорректныеПроводки».

Англоязычный синоним ключевого слова `КорректныеПроводки` — `CorrectEntries`.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
```

Атрибуты объекта «КорректныеПроводки»

Комментарий

Описание корректной проводки.

Синтаксис:

```
Комментарий
```

Англоязычный синоним:

```
Description
```

Описание:

Атрибут `Комментарий` дает доступ к значению описания корректной проводки.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Сообщить(КП.Комментарий);
КонецЦикла;
```

СчетДт

Счет дебета корректной проводки.

Синтаксис:

```
СчетДт
```

Англоязычный синоним:

```
AccountDt
```

Описание:

Атрибут `СчетДт` дает доступ к значению счета дебета корректной проводки.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Сообщить(КП.Комментарий + " счет дебета — " + КП.СчетДт);
КонецЦикла;
```

СчетКт

Счет кредита корректной проводки.

Синтаксис:

```
СчетКт
```

Англоязычный синоним:

```
AccountKt
```

Описание:

Атрибут `СчетКт` дает доступ к значению счета кредита корректной проводки.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Сообщить(КП.Комментарий + " счет кредита — " + КП.СчетКт);
КонецЦикла;
```

Методы объекта «КорректныеПроводки»

Выбрана

Проверить факт: выбрана ли корректная проводка.

Синтаксис:

```
Выбрана()
```

Англоязычный синоним:

```
Selected
```

Возвращаемое значение:

Число: 1 — если корректная проводка выбрана; 0 — если не выбрана.

Описание:

Метод `Выбрана` возвращает число со значением 1 — объект выбран или 0 — если не выбран.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Если Кп.Выбрана() = 1 Тогда
      Сообщить(КП.Комментарий + " счет кредита — " + КП.СчетКт);
   КонецЕсли;
КонецЦикла;
```

ВыбратьКорректныеПроводки

Открыть выборку корректных проводок по плану счетов.

Синтаксис:

```
ВыбратьКорректныеПроводки(<ПланСчетов>)
```

Англоязычный синоним:

```
SelectCorrectEntries
```

Параметры:

`<ПланСчетов>`	Необязательный параметр. Значение типа «План счетов». Если не задан, то по всем.

Возвращаемое значение:

Число: 1 — если есть хотя бы одна корректная проводка, удовлетворяющая условию; 0 — нет ни одной корректной проводки.

Описание:

Метод `ВыбратьКорректныеПроводки` предоставляет возможность выбирать корректные проводки (открывает выборку) при помощи метода `ПолучитьКорректнуюПроводку`.

Дальнейшая выборка при помощи метода `ПолучитьКорректнуюПроводку` будет происходить среди корректных проводок указанного плана счетов.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Если КП.Выбрана() = 1 Тогда
      Сообщить(КП.Комментарий + " счет кредита — " + КП.СчетКт);
   КонецЕсли;
КонецЦикла;
```

ВыбратьКорректныеПроводкиПоСчету

Открыть выборку корректных проводок для указанного счета.

Синтаксис:

```
ВыбратьКорректныеПроводкиПоСчету(<Счет>, <ТипСчета>, <ПланСчетов>)
```

Англоязычный синоним:

```
SelectCorrectEntriesByAccount
```

Параметры:

`<Счет>`	Код счета или сам счет для которого отбирать корректные проводки.
`<ТипСчета>`	Необязательный параметр. Число: 0 — переданный счет является счетом дебета; 1 — переданный счет является счетом кредита. Значение по умолчанию 0.
`<ПланСчетов>`	Необязательный параметр. Значение типа «План счетов». Значение по умолчанию — главный план счетов.

Возвращаемое значение:

Число: 1 — если есть хотя бы одна корректная проводка, удовлетворяющая условию; 0 — нет ни одной корректной проводки.

Описание:

Метод `ВыбратьКорректныеПроводкиПоСчету` предоставляет возможность выбирать корректные проводки (открывает выборку) при помощи метода `ПолучитьКорректнуюПроводку`.

Дальнейшая выборка при помощи метода `ПолучитьКорректнуюПроводку` будет происходить среди корректных проводок указанного плана счетов.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводкиПоСчету("41.1");
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Если Кп.Выбрана() = 1 Тогда
      Сообщить(КП.Комментарий + " счет кредита — " + КП.СчетКт);
   КонецЕсли;
КонецЦикла;
```

ПолучитьКорректнуюПроводку

Получить из выборки следующую корректную проводку.

Синтаксис:

`ПолучитьКорректнуюПроводку()`

Англоязычный синоним:

`GetCorrectEntry`

Возвращаемое значение:

Число: 1 — корректная проводка получена; 0 — не получена (отсутствует).

Описание:

Метод `ПолучитьКорректнуюПроводку` выбирает следующую корректную проводку в выборке, открытой перед этим при помощи метода `ВыбратьКорректныеПроводки` или `ВыбратьКорректныеПроводкиПоСчету`.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПолучитьКорректнуюПроводку() = 1 Цикл
   Если Кп.Выбрана() = 1 Тогда
      Сообщить(КП.Комментарий + " счет кредита — " + КП.СчетКт);
   КонецЕсли;
КонецЦикла;
```

Новая

Добавить новую корректную проводку.

Синтаксис:

`Новая()`

Англоязычный синоним:

`New`

Описание:

Метод `Новая` добавляет новую корректную проводку.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.Новая();
КП.Комментарий = "НДС с комисс. вознаграждения";
КП.СчетДт = "46.28";
КП.СчетКт = "68.2";
Кп.Записать();
```

Записать

Записать измененную или новую корректную проводку.

Синтаксис:

`Записать()`

Англоязычный синоним:

`Write`

Возвращаемое значение:

Число: 1 — корректная проводка успешно записана; 0 — не записана.

Описание:

Метод `Записать` записывает измененную или новую корректную проводку.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.Новая();
КП.Комментарий ="НДС с комисс. вознаграждения";
КП.СчетДт = "46.28";
КП.СчетКт = "68.2";
Кп.Записать();
```

Удалить

Удалить корректную проводку.

Синтаксис:

`Удалить()`

Англоязычный синоним:

Delete

Возвращаемое значение:

Число: 1 — корректная проводка успешно удалена; 0 — не удалена.

Описание:

Метод Удалить удаляет корректную проводку.

Пример:

```
КП = СоздатьОбъект("КорректныеПроводки");
КП.ВыбратьКорректныеПроводки();
Пока КП.ПояучитьКорректнуюПроводку() = 1 Цикл
   Если КП.Комментарий = "НДС с комисс. вознаграждения" Тогда
      КП.Удалить();
      Прервать;
   КонецЕсли;
КонецЦикла;
```

Глава 22
Работа с Журналами расчетов

Журнал расчетов — это средство для просмотра и редактирования результатов расчета. Каждая строка журнала отражает единичное событие расчета для того или иного объекта. Такие события называются «расчетами» и характеризуются следующими данными:

- объект, для которого произведен расчет;
- вид (т. е. способ) расчета;
- документ, на основании которого введен этот расчет;
- результат расчета;
- время действия (т. е. расчет имеет дату начала и дату окончания). период регистрации, во время которого расчет введен в журнал (это понятие не совпадает с понятием времени действия, так как время действия акта расчета и период регистрации, во время которого он введен в систему, могут не совпадать)

Справочник объектов расчета. Одним из самых важных свойств журнала расчетов является ссылка на справочник, элементы которого являются объектами расчета. Справочник объектов расчета может быть как простым, так и иерархическим. Не обязательно все элементы этого справочника должны рассчитываться конкретным журналом расчетов, но, в свою очередь, все строки журнала расчетов должны соответствовать тому или иному элементу справочника объектов расчета. Как правило, журнал расчетов содержит несколько записей по одному объекту расчета, но одна запись журнала расчетов не может соответствовать сразу нескольким объектам.

Расчетный период. Журнал расчетов имеет определенную периодичность, т. е. все расчеты в нем выполняются в пределах определенного временного интервала. Каждая запись журнала расчетов, соответствующая одному акту расчета, также имеет временное протяжение (т. е. имеет дату начала и дату окончания).

Важно! Временной интервал каждой отдельной записи журнала не может лежать в разных расчетных периодах журнала.

Записи, лежащие в текущем расчетном периоде, могут, тем не менее, иметь период действия, не принадлежащий текущему расчетному периоду. Это значит, что даты начала и окончания конкретной записи могут «выпадать» из текущего периода расчета и относиться к одному из прошлых или будущих периодов. Самым очевидным примером здесь может служить перерасчет зарплаты сотрудника за прошлый период или начисления будущего периода.

Примеры. Журнал расчета заработной платы сотрудников предприятия (каждый сотрудник — объект расчета), при этом выполняется расчет тех или иных начислений и удержаний (виды расчета), имеющих определенную продолжительность и результат. Другой пример — журнал расчета амортизации основных средств, где справочником объектов расчета является справочник основных средств предприятия, а видами расчета —алгоритм расчета амортизации, списания, переоценки и т. д. Третий пример — журнал расчета дивидендов для акционеров АО, где объекты расчета — список акционеров предприятия, а виды расчета — способы начислений дивидендов на акции разных типов.

Контекст работы с журналом расчета

Так же, как и при работе с другими данными системы 1С:Предприятие, доступ к атрибутам журнала расчетов и вызов его методов зависит от контекста выполнения модуля программы. В контексте выполнения расчета (см. «Виды программных модулей»), доступны атрибуты, методы журнала расчетов, т. е. они пишутся непосредственно, с указанием необходимых параметров.

Пример:

```
// Запись рассчитанного значения в атрибут Результат журнала расчетов
Результат = Объект.Оклад * Дни / ВсегоДней;
```

Во всех остальных случаях доступ к атрибутам и методам журнала расчетов производится через переменную, являющуюся ссылкой на объект типа «Журнал расчетов», т. е. созданную функцией `СоздатьОбъект` с ключевым словом `ЖурналРасчетов`.

Англоязычный синоним ключевого слова `ЖурналРасчетов` — `CalcJournal`.

Пример:

```
// Рассчитаем все записи журнала расчетов Зарплата
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьПериод(НачалоТекущегоПериода());
Пока ПолучитьЗапись() = 1 Цикл
   ЖР.Рассчитать();
КонецЦикла;
```

Контекст работы с записями журнала расчетов

При работе с журналом расчетов приходится сталкиваться с понятием «записи журнала расчетов». Ссылкой на объект типа «запись журнала расчетов» является, например, атрибут `ПервичнаяЗапись` журнала расчетов. Метод журнала `ТекущаяЗапись` также возвращает объект такого типа.

Кроме того, объект этого типа передается как параметр предопределенной процедуре формы журнала расчетов `ПриРасчете` в том случае, если выполняется расчет одной записи журнала расчетов. С точки зрения встроенного языка, ссылки на записи журнала расчетов — агрегатные объекты, имеющие свои атрибуты.

При работе с переменными или параметрами, представляющими собой ссылки на записи журнала расчетов, можно использовать любые атрибуты, присущие журналу расчетов, как агрегатному объекту (см. ниже).

Запись журнала расчетов как агрегатный объект не имеет методов.

Пример:

```
Процедура ПриРасчете(Объект)
   // определим, что считаем?
   Если ТипЗначения(Объект) = 12 Тогда
      // имеем дело с документом
      Если Объект.Автор <> Расчетчик Тогда
         // если автор данного документа не есть текущий расчетчик
         // запретим расчет чужого документа!
         СтатусВозврата(0);
      КонецЕсли;
   ИначеЕсли ТипЗначения(Объект) = 11 Тогда
      // а это — элемент справочника
      Если Объект.Владелец <> Расчетчик Тогда
         // если рассчитываемый объект лежит в папке текущего
         // расчетчика — считать можно, иначе — нет
         СтатусВозврата(0);
      КонецЕсли;
   Иначе
      // иначе — считаем одну запись
      Если (Объект.Документ.Автор <> Расчетчик) ИЛИ
           (Объект.Объект.Владелец <> Расчетчик) Тогда
         // не дадим считать запись чужого документа или объекта
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

Контекст работы с периодом журнала расчетов

При работе с журналом расчетов приходится сталкиваться с понятием «период журнала расчетов». Ссылкой на такого рода объект являются, например, атрибуты `ПериодРегистрации` и `ПериодДействия` журнала расчетов. Кроме того, объект этого типа передается в качестве одного из параметров предопределенной процедуре глобального модуля `ПриСменеРасчетногоПериода`. С точки зрения встроенного языка, период журнала расчетов — агрегатный объект, имеющий свои атрибуты (см. ниже) и методы. Атрибуты периода журнала расчетов предназначены только для чтения.

Атрибуты периода журнала расчетов

ДатаНачала

Дата начала периода журнала расчетов.

Синтаксис:

```
ДатаНачала
```

Англоязычный синоним:

```
DateFrom
```

Описание:

Атрибут типа «дата» — дата начала периода журнала расчетов.

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: `ДатаОкончания`

ДатаОкончания

Дата окончания периода журнала расчетов.

Синтаксис:

```
ДатаОкончания
```

Англоязычный синоним:

```
DateTill
```

Описание:
Атрибут типа «дата» — дата окончания периода журнала расчетов.
См. также: ДатаНачала

ОписательПериода

Возвращает строку-описатель периода.
Синтаксис:
ОписательПериода
Англоязычный синоним:
PeriodDescriptor
Описание:
Атрибут ОписательПериода является строковым представлением расчетного периода журнала расчетов. Тип формируемой строки различный для разной периодичности журнала расчетов. Например, для периода журналов с месячной периодичностью строка имеет вид "Январь 1997г", в случае квартальной периодичности журнала расчетов — "1 Квартал 1997г", а в случае недельной периодичности — "12.02.97 - 18.02.97".
Как правило, применяется при формировании отчетов.
Пример:

```
Процедура ВыводПоПериодам()
   Перем Запрос, ТекстЗапроса, Таб;
   Перем ДатаАкт;
   ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   ДатаАкт = ЖЗ.НачалоТекущегоПериода();
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(ВыводПоПериодам)
   |Период с ДатаАкт по ДатаАкт;
   |Сотр = ЖурналРасчетов.Зарплата.Объект;
   |Пер = ЖурналРасчетов.Зарплата.ПериодРегистрации;
   |Рез = ЖурналРасчетов.Зарплата.Результат;
   |Группировка Сотр упорядочить по Сотр.МестоРаботы без групп;
   |Группировка Пер;
   |Функция Сум = Сумма(Рез);
   |"//}}ЗАПРОС
   ;
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
      Возврат;
   КонецЕсли;
   Пока Запрос.Группировка("Сотр") = 1 Цикл
      Сообщить(Запрос.Сотр.Наименование);
      Пока Запрос.Группировка("Пер") = 1 Цикл
         Сообщить(Запрос.Пер.ОписательПериода + "-" + Запрос.Сум);
      КонецЦикла;
   КонецЦикла;
КонецПроцедуры
```

Методы периода журнала расчетов

ПрибавитьПериод

Возвращает очередной период журнала на базе текущего периода.
Синтаксис:
ПрибавитьПериод(<Количество>)
Англоязычный синоним:
AddPeriod
Параметры:

<Количество>	Число периодов, на которое отстоит текущий период от искомого. По умолчанию — 1 (т. е. метод возвращает следующий период).

Возвращаемое значение:
Период журнала, отстоящий от заданного на определяемое параметром <Количество> число периодов.
Описание:
Метод предназначен для получения периода отстоящего от текущего на заданное число периодов. Если <Количество> положительное число, возвращаемое значение — будущий период, в противном случае — прошлый период.
Пример:

```
ЖР = СоздатьОбъект{"ЖурналРасчетов.Зарплата");
ЖР.ВыбратьПериод(РабочаяДата());
//на три расчетных периода (месяца) назад
Пер = ЖР.ПериодДействия.ПрибавитьПериод(-3);
```

```
// теперь отбираем записи по нужному нам периоду
ЖР.ВыбратьПериод(Пер.ДатаНачала);
Всего = 0;
Пока ПолучитьЗапись() = 1 Цикл
   Если ЖР.ВидРасч = ВидРасчета.Доплата Тогда
      Всего = Всего + ЖР.Результат;
   КонецЕсли;
КонецЦикла;
```

Атрибуты журнала расчетов

Документ

Документ-основание текущей записи журнала расчетов.

Синтаксис:

Документ

Англоязычный синоним:

Document

Описание:

Атрибут типа «Документ». Ссылка на документ, на основании которого данная запись (расчет) введена в журнал расчетов. Это может быть, например, документ «Больничный лист», породивший одну или несколько записей с видом расчета «Оплата больничного листа».

Этот атрибут журнала расчетов заполняется в момент проведения документа ссылкой на этот документ, в случае использования методов ВвестиРасчет, ВвестиПерерасчет, ЗаписатьРасчет, или ссылкой на другой документ, в случае использования методов ВвестиРасчетНаОсновании, ЗаписатьРасчетНаОсновании, ВвестиПерерасчетНаОсновании (см. ниже).

Атрибут Документ предназначен только для чтения.

Пример:

```
Процедура ПровестиРасчет()
   Календ = Объект.Категория.Получить(ДатаНачала).Календарь;
   Дней = Календ.Дней(ДатаНачала, ДатаОкончания);
   Если ОткрытДок(Документ) = 0 Тогда
      РассчитатьСреднюю(Документ, Константа.МесСреднБЛ,
           ГруппаРасчетов.СредняяДляБЛ, 0);
   КонецЕсли;
   // среднюю зарплату берем из документа породившего расчет
   Результат =Окр(Дней * Документ.СрЗарплата);
   Дни = ?(Сторно = 1, -Дней, Дней);
КонецПроцедуры
```

См. также: ВвестиРасчет, ЗаписатьРасчет, ВвестиПерерасчет, ЗаписатьРасчетНаОсновании, ВестиРасчетНаОсновании, ВвестиПерерасчетНаОсновании

РодительскийДокумент

Документ, который ввел данную запись в журнал расчетов.

Синтаксис:

РодительскийДокумент

Англоязычный синоним:

ParentDocument

Описание:

Атрибут типа «Документ». Ссылка на документ расчета, который ввел данную запись (расчет) журнала расчетов.

В момент проведения документа, при вводе новых записей в журнал расчетов любым способом, атрибут РодительскийДокумент заполняется ссылкой на тот документ, который проводится.

Атрибут РодительскийДокумент предназначен только для чтения.

Пример:

```
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьЗаписиПоОбъекту();
Пока ЖР.ПолучитьЗапись() = 1 Цикл
   Если (ЖР.ВидРасч = ВидРасчета.ПоОкладу) И
            (ЖР.РодительскийДокумент.Вид() <> "НачалоМесяца") Тогда
      СообщитьОПерерасчете();
   КонецЕсли;
КонецЦикла;
```

См. также: ВвестиРасчет, ЗаписатьРасчет, ВвестиПерерасчет, ЗаписатьРасчетНаОсновании, ВвестиРасчетНаОсновании, ВвестиПерерасчетНаОсновании

Объект

Элемент справочника, для которого введена данная запись журнала расчетов.

Синтаксис:
Объект
Англоязычный синоним:
Object
Описание:
Атрибут типа «Справочник». Ссылка на конкретный элемент того справочника, который при конфигурировании журнала расчетов задан как список объектов расчета (справочник объектов расчета). К этому конкретному элементу справочника имеет отношение данная запись (расчет) журнала расчетов.
Атрибут Объект предназначен только для чтения.
Пример:
```
// Расчет профвзносов
Процедура ПровестиРасчет()
   // объект расчета — сотрудник
   Если Объект.Профсоюз.Получить(ДатаОкончания) = Да Тогда
      // Зануляем результат перед расчетом для того,
      // чтобы не учитывать его при расчете уже взятого налога!!!
      Результат = 0;
      // Журнал расчетов
      ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
      ЖР.ВыбратьЗаписиПоОбъекту(Объект, ЖР.НачалоПериодаПоДате(ДатаНачала),
            ЖР.КонецПериодаПоДате(ДатаНачала));
      Облагаем = 0;
      УжеВзяли = 0;
      Группа = ГруппаРасчетов.ОблагаетсяПроф;
      Пока ЖР.ПолучитьЗапись() > 0 Цикл
         Если ЖР.ВидРасч.ВходитВГруппу(Группа) = 1 Тогда
            Облагаем = Облагаем + ЖР.Результат;
         ИначеЕсли ЖР.ВидРасч = ВидРасчета.ПрофВзносы Тогда
            УжеВзяли = УжеВзяли + ЖР.Результат;
         КонецЕсли;
      КонецЦикла;
      Результат = Окр(Константа.ПроцентПроф.Получить(ДатаНачала) / 100 *
            Облагаем - УжеВзяли);
   Иначе
      Результат = 0;
   КонецЕсли;
КонецПроцедуры
```
См. также: ВыбратьПериодПоОбъекту, ВыбратьЗаписиПоОбъекту

ВидРасч

Вид расчета текущей записи журнала расчетов.
Синтаксис:
ВидРасч
Англоязычный синоним:
CalculKind
Описание:
Атрибут типа «ВидРасчета». Ссылка на конкретный вид расчета, алгоритм которого будет использован при вычислении результата данной записи журнала расчетов (акта расчета).
Атрибут ВидРасч предназначен только для чтения.
Пример:
```
// Перечисление в банк
Процедура ПровестиРасчет()
   // Процедура выполняется при проведении расчета
   // посчитаем сумму на руки
   НаРуки = 0;
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   Пока ЖР.ПолучитьЗапись() = 1 Цикл
      // если строка журнала расчетов имеет вид расчета,
      // входящий в группу "Все начисления"
      Если ЖР.ВидРасч.ВходитВГруппу(Группа.ВсеНачислнения) Тогда
         НаРуки = НаРуки + ЖР.Результат;
         // ...или "Все удержания"
      ИначеЕсли ЖР.ВидРасч.ВходитВГруппу(Группа.ВсеУдержания) Тогда
         НаРуки = НаРуки - ЖР.Результат;
      КонецЕсли;
   КонецЦикла;
   Если Документ.Сумма <> 0 Тогда
      Результат = Окр(Мин(Документ.Сумма, НаРуки));
```

```
    ИначеЕсли Документ.Процент <> 0 Тогда
       НаРуки = 0;
       Результат = Окр(НаРуки * Процент / 100);
    КонецЕсли;
    Если Документ.Банк.Выбран() Тогда
       ПроцентУд = Документ.Банк.ПроцентУдержания;
    Иначе
       Сообщение("Не указан банк для перечисления средств!");
       Возврат;
    КонецЕсли;
    ЖР.ВвестиРасчет(Объект, ВидРасчета.БанковскиеИздержки,
                  ДатаНачала, ДатаОкончания, Окр(Результат * ПроцентУд / 100));
КонецПроцедуры
```

ДатаНачала

Дата начала действия записи журнала расчетов.

Синтаксис:

`ДатаНачала`

Англоязычный синоним:

`DateFrom`

Описание:

Атрибут типа «дата» — дата начала действия текущего акта расчета (записи журнала расчетов). Эта дата может не совпадать с периодом регистрации Атрибут `ДатаНачала` предназначен только для чтения.

Пример:

```
// процедура выполняется в контексте журнала расчетов
// ДатаНачала и ДатаОкончания доступны непосредственно!
// календарь
Календ = Объект.Категория.Получить(ДатаНачала).Календарь;
// размер оклада
Тариф = Объект.Тариф.Получить(ДатаОкончания);
Часов = Календ.Часов(ДатаНачала.ДатаОкончания);
Результат = Окр(Тариф*Часов);
Часы = ?(Сторно = 1, -Часов, Часов);
```

См. также: `ДатаОкончания`, `ПериодДействия`, `ПериодРегистрации`

ДатаОкончания

Дата окончания действия записи журнала расчетов.

Синтаксис:

`ДатаОкончания`

Англоязычный синоним:

`DateTill`

Описание:

Атрибут типа «дата» — дата окончания действия текущего акта расчета (записи журнала расчетов).

Атрибут `ДатаОкончания` предназначен только для чтения.

Пример:

См. предыдущий пример.

См. также: `ДатаНачала`, `ПериодДействия`, `ПериодРегистрации`

ПериодДействия

Период действия записи журнала расчетов.

Синтаксис:

`ПериодДействия`

Англоязычный синоним:

`EffectivePeriod`

Описание:

Атрибут типа «период журнала расчетов» — период действия текущей записи журнала расчетов. Под периодом действия подразумевается тот расчетный период журнала, в который попадают `ДатаНачала` и `ДатаОкончания` действия записи журнала расчетов (акта расчета).

Атрибут `ПериодДействия` предназначен только для чтения.

Пример:

```
Процедура ПоПодр()
   Перем Запрос, ТекстЗапроса, Таб;
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(ПоПодр)
   |Период с ДатаНач по ДатаКон;
   |Док = ЖурналРасчетов.Зарплата.ТекущийДокумент;
```

```
|ПД = ЖурналРасчетов.Зарплата.ПериодДействия;
|Подр = ЖурналРасчетов.Зарплата.Объект.МестоРаботы.Владелец;
|Рез = ЖурналРасчетов.Зарплата.Результат;
|Группировка ПД;
|Группировка Подр без групп;
|Функция Итог = Сумма(Рез);
|"//}}ЗАПРОС
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
// Подготовка к заполнению выходных форм данными запроса
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("ПоПодр");
// Заполнение полей "Заголовок"
Таб.ВывестиСекцию("Заголовок");
Пока Запрос.Группировка("ПД") = 1 Цикл
   // Заполнение полей ПД
   Таб.ВывестиСекцию("ПД");
   Сообщение(Запрос.ПД.Описатель);
   Пока Запрос.Группировка("Подр") = 1 Цикл
      // Заполнение полей Подр
      Таб.ВывестиСекцию("Подр");
   КонецЦикла;
КонецЦикла;
// Заполнение полей "Итого"
Таб.ВывестиСекцию("Итого");
// Вывод заполненной формы
Таб.Опции(1, 0, 1, 0);
Таб.Показать("ПоПодр", "");
КонецПроцедуры
```

См. также: ДатаНачала, ПериодДействия, ПериодРегистрации

ПериодРегистрации

Период регистрации записи журнала расчетов.

Синтаксис:

ПериодРегистрации

Англоязычный синоним:

RegistrationPeriod

Описание:

Атрибут типа «период журнала расчетов» — расчетный период, в котором текущая запись была введена в журнал расчетов. Это тот расчетный период, который на момент ввода данной записи был установлен для журнала расчетов как текущий.

Атрибут ПериодРегистрации предназначен только для чтения.

Пример:

См. предыдущий пример.

См. также: ДатаНачала, ДатаОкончания, ПериодДействия

Сторно

Признак сторнирующей записи.

Синтаксис:

Сторно

Англоязычный синоним:

Storno

Описание:

Атрибут типа «число», принимает два значения: 1 — для сторнирующих записей журнала расчетов и 0 — для обычных записей. Признак «сторно» равен 1 не только для простых сторно-записей, но и для рассчитанных, отредактированных вручную или зафиксированных (не подлежащих редактированию) сторно-записей.

Сторнирующие записи могут появляться в журнале расчетов в результате выполнения методов ВвестиРасчет, ЗаписатьРасчет и им подобных, если при этом затрагиваются один или несколько прошлых расчетных периодов и вводимый вид расчета является «вытесняющим». При этом система автоматически вводит необходимые сторнирующие записи. Для таких записей (введенных системным образом) признак сторнирования не может быть переопределен.

Если сторнирующая запись введена программным образом, т. е. атрибут Сторно задан за счет применения метода УстановитьРеквизит или непосредственным присвоением:

```
ЖрнРасчета = СоздатьОбъект("ЖурналРасчета.Зарплата");
ЖрнРасчета.Сторно = 1;
```

тогда он может быть переопределен программным образом.

Пример:

```
*
Процедура ПровестиРасчет()
   // календарь
   Календ = Объект.Категория.Получить(ДатаОкончания).Календарь;
   // размер оклада
   Оклад = Объект.Оклад.Получить(ДатаОкончания);
   Дней = Календ.Дней(ДатаНачала, ДатаОкончания);
   ВсегоДней = Календ.Дней(НачалоПериодаПоДате(ДатаНачала),
               КонецПериодаПоДате(ДатаНачала));
   Если ВсегоДней > 0 Тогда
      Результат = Окр(Оклад * Дней / ВсегоДней);
      // если запись-сторно — поставим отрицательные дни!
      Дни = ?(Сторно = 1, -Дней, Дней);
   Иначе
      Результат = 0;
      Дни = 0;
      Сообщить("Неправильный календарь!");
   КонецЕсли;
КонецПроцедуры
*
// вводим сторнирующие рассчитанные записи
ВР = ВидРасчета.ДоплатаКОкладу;
ЖрнЗарплата.УстановитьРеквизит("Сторно", 1);
ЖрнЗарплата.УстановитьРеквизит("Рассчитана", 1);
ЖрнЗарплата.ВвестиРасчет(Сотрудник, ВР, , , Сумма * Процент);
```

См. также: УстановитьРеквизит, ВвестиРасчет, ЗаписатьРасчет, Рассчитана

Рассчитана

Признак того, что запись рассчитана.

Синтаксис:

Рассчитана

Англоязычный синоним:

Calculated

Описание:

Атрибут типа «число», принимает два значения: 1 — для рассчитанных записей журнала расчетов и 0 — для нерассчитанных записей.

Запись журнала расчетов становится рассчитанной в результате удачного выполнения одной из соответствующих команд меню «Действия» или при выполнении методов журнала расчетов Рассчитать, ВыпролнитьРасчет.

Атрибут Рассчитана предназначен только для чтения.

Пример:

```
// Перед выводом отчета проверим, все ли записи рассчитаны
ЖрнЗарплата = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖрнЗарплата.ВыбратьПериодПоОбъекту(Сотрудник, ДатаОтчета);
// признак нерассчитанности
ПлохоДело=0;
Пока ЖрнЗарплата.ПолучитьЗапись() = 1 Цикл
   Если ЖрнЗарплата.Рассчитана = 0 Тогда
      ПолохоДело = 1;
      Прервать;
   КонецЕсли;
КонецЦикла;
Если ПлохоДело = 1 Тогда
   Предупреждение("Не проведен полный расчет сотрудника!
                  | Формирование отчета невозможно");
   Возврат;
КонецЕсли;
// продолжим формирование документа
```

См. также: Исправлена, Фиксирована

Исправлена

Признак того, что запись исправлена вручную.

Синтаксис:

Исправлена

Англоязычный синоним:

Updated

Описание:

Атрибут типа «число», принимает два значения: 1 — для записей журнала расчетов, результат которых исправлен «вручную» и 0 — для остальных записей.

Запись журнала расчетов становится исправленной при редактировании результата расчета непосредственно в журнале расчетов. Исправленные вручную записи не редактируются при очередном сеансе расчета, т. е. результат автоматического расчета не заменяет результат ручного ввода. Это значит, что результат ручного редактирования «главнее» результата автоматического расчета.

Атрибут Исправлена предназначен только для чтения.

Пример:

```
// Проверим, не редактировали ли результаты ЖрнЗарплата = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖрнЗарплата.ВыбратьЗаписиПоДокументу(ТекущийДокумент());
Счетчик = 0;
Пока ЖрнЗарплата.ПолучитьЗапись() = 1 Цикл
   Если ЖрнЗарплата.Исправлена = 1 Тогда
      Счетчик = Счетчик + 1;
   КонецЕсли;
КонецЦикла;
Если Счетчик <> 0 Тогда
   Сообщение("Исправленных записей — " + Счетчик);
КонецЕсли;
// продолжим формирование документа
```

См. также: Рассчитана, Фиксирована

Фиксирована

Признак того, что результат расчета записи защищен от исправления.

Синтаксис:

Фиксирована

Англоязычный синоним:

Locked

Описание:

Атрибут типа «число», принимает два значения: 1 — для фиксированных записей журнала расчетов и 0 — для остальных записей.

Запись журнала расчетов становится фиксированной при выполнении метода журнала расчетов ФиксироватьЗапись. Признак фиксированности записи можно снять, применив метод журнала расчетов ОсвободитьЗапись.

После фиксации записи результат ее расчета не может быть изменен никаким, в том числе, программным способом.

Атрибут Фиксирована предназначен только для чтения.

Пример:

```
Если ЖрнЗарплата.Фиксирована = 1 Тогда
   Если ДатаДок > ЖрнЗарплата.НачалоТекущегоПериода() Тогда
      ОсвободитьЗапись();
   Иначе
      // отказ от расчета;
      // ....
   КонецЕсли;
КоненцЕсли;
```

См. также: ФиксироватьЗапись, ОсвободитьЗапись

Перерасчет

Признак того, что запись является перерасчетом другой записи прошлого периода.

Синтаксис:

Перерасчет

Англоязычный синоним:

Recalc

Описание:

Атрибут типа «число», принимает два значения: 1 — для записей-перерасчетов и 0 — для остальных записей.

Записи-перерасчеты вводятся в журнал расчетов при выполнении одного из методов журнала расчетов: ВвестиПерерасчет или ВвестиПерерасчетНаОсновании.

Атрибут Перерасчет предназначен только для чтения.

Пример:

```
ЖрнЗарплата = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖрнЗарплата.ВыбратьПериодПоОбъекту(Сотрудник, ДатаОтчета);
Есть = 0;
Пока ЖрнЗарплата.ПолучитьЗапись() = 1 Цикл
   Если ЖрнЗарплата.Перерасчет = 0 Тогда
      Есть = 1;
      Пер = ЖрнЗарплата.ПервичнаяЗапись.ПериодРегистрации;
```

```
      Прервать;
   КонецЕсли;
КонецЦикла;
Если Есть = 1 Тогда
   Предупреждение("За истекший период проводились перерасчеты
          | прошлого периода " + Пер.Описатель);
   Возврат;
КонецЕсли;
// продолжим формирование документа
```

См. также: ПервичнаяЗапись

ПервичнаяЗапись

Первичная запись записи-перерасчета. Запись, на основании которой введена текущая запись-перерасчет.

Синтаксис:

ПервичнаяЗапись

Англоязычный синоним:

ParentRecord

Описание:

Атрибут типа «запись журнала расчетов». Имеет непустое значение для записей-перерасчетов, т. е. тех записей, которые введены в систему одним из методов журнала расчетов: ВвестиПерерасчет или ВвестиПерерасчетНаОсновании.

Атрибут ПервичнаяЗапись предназначен только для чтения.

Пример:

См. предыдущий пример.

См. также: Перерасчет, ВвестиПерерасчет, ВвестиПерерасчетНаОсновании

Результат

Результат расчета,

Синтаксис:

Результат

Англоязычный синоним:

Result

Описание:

Атрибут типа «число». Используется для доступа к результату расчета записи. Как правило, самым важным действием процедуры ПровестиРасчет модуля расчета является вычисление результата расчета и заполнение атрибута Расчет.

Пример:

```
*
// Переберем все записи текущего периода журнала расчетов
//по текущему основному средству и просуммируем результат
ЖР = СоздатьОбъект("ЖурналРасчетов.Амортизация");
ЖР.ВыбратьПериодПоОбъекту(Объект, ЖР.НачалоТекущегоПериода());
// Инициализируем переменную
Сумма = 0;
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   Сумма = Сумма + ЖР.Результат;
КонецЦикла
*
Результат = Оклад * Дней / ОтработаноДней;
```

<Реквизит>

Значение реквизита журнала расчетов.

Синтаксис:

<Реквизит> Идентификатор реквизита журнала расчетов, как он задан в конфигураторе.

Описание:

Атрибут <Реквизит> предоставляет доступ к значению реквизита записи журнала расчетов. В тексте программного модуля в качестве названия реквизита используется идентификатор конкретного реквизита журнала расчетов, созданного в конфигураторе.

Пример:

```
//В этом примере журнал расчетов "Зарплата"
// имеет реквизит "ХозОперация"
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьПериод(ЖР.НачалоТекущегоПериода());
Пока ЖР.ПолучитьЗапись() = 1 Цикл
   ЖР.ХозОперация = ПолучитьХО(ЖР.ТекущаяЗапись());
   // вызвали вспомогательную функцию пользователя: ПолучитьХО
КонецЦикла;
```

Методы журнала расчетов

НачалоТекущегоПериода

Возвращает дату начала текущего расчетного периода.

Синтаксис:

```
НачалоТекущегоПериода()
```

Англоязычный синоним:

```
CurrentPeriodBegin
```

Возвращаемое значение:

Значение типа «дата» — начало текущего расчетного периода журнала расчетов.

Описание:

Возвращает дату начала текущего периода журнала расчетов. Текущий период — отрезок времени, обсчитываемый журналом расчетов в данный момент. Величина расчетного периода журнала расчетов определяется при конфигурировании журнала и может принимать значения: день, неделя, месяц, квартал, год.

Текущий период журнала расчетов устанавливается при выполнении операции «Установить период расчета» меню «Действия» журнала расчетов. В случае, если периодичность журнала расчетов равна дню, методы `НачалоТекущегоПериода` и `КонецТекущегоПериода` возвращают одно и то же значение.

Пример:

```
//
ЖРЗарп = СоздатьОбъект("ЖурналРасчета.Зарплата");
Начало = ЖРЗарп.НачалоТекущегоПериода();
```

См. также: `КонецТекущегоПериода`, `ПериодДействия`, `ПериодРегистрации`

КонецТекущегоПериода

Возвращает дату окончания текущего расчетного периода.

Синтаксис:

```
КонецТекущегоПериода();
```

Англоязычный синоним:

```
CurrentPeriodEnd
```

Возвращаемое значение:

Значение типа «дата» — окончание текущего расчетного периода журнала расчетов.

Описание:

Возвращает дату окончания текущего расчетного периода журнала расчетов. Текущий расчетный период — это отрезок времени, обсчитываемый журналом расчетов в настоящее время. См. описание метода `НачалоТекущегоПериода`.

Пример:

```
// ЖРЗарп = СоздатьОбъект("ЖурналРасчета.Зарплата");
Календ = СоздатьОбъект("Календарь.Служащие");
НормаОтработки = Календ.Дней(ЖРЗарп.НачалоТекущегоПериода(),
                             ЖРЗарп.КонецТекущегоПериода());
```

См. также: `НачалоТекущегоПериода`, `ПериодДействия`, `ПериодРегистрации`

НачалоПериодаПоДате

Возвращает дату начала произвольного расчетного периода.

Синтаксис:

```
НачалоПериодаПоДате(<Дата>)
```

Англоязычный синоним:

```
PeriodBeginByDate
```

Параметры:

`<Дата>`	Любая дата, которая попадает в требуемый период.

Возвращаемое значение:

Значение типа «дата» — начало расчетного периода журнала расчетов.

Описание:

Метод позволяет определить дату начала того периода журнала расчетов, в который попадает заданная `<Дата>`. Работа метода не зависит от наличия в журнале расчетов данных за соответствующий расчетный период.

Пример:

```
// В каждой строке журнала расчетов проставим в реквизит Дни
// количество банковских дней того периода расчета, в который
// попадает запись
ЖР = СоздатьОбъект("ЖурналРасчетов.Дивиденды");
Календ = СоздатьОбъект("Календарь.РаботаБанка");
ЖР.ВыбратьПериод(ЖР.НачалоТекущегоПериода());
Пока ЖР.ПолучитьЗапись() = 1 Цикл
   ЖР.Дни = Календ.Дней(НачалоПериодаПоДате(ЖР.ДатаНачала),
                        КонецПериодаПоДате(ЖР.ДатаНачала));
КонецЦикла;
```

См. также: `КонецПериодаПоДате`

КонецПериодаПоДате

Возвращает дату окончания произвольного расчетного периода.

Синтаксис:

КонецПериодаПоДате(<Дата>)

Англоязычный синоним:

PeriodEndByDate

Параметры:

<Дата>	Любая дата, которая попадает в требуемый период.

Возвращаемое значение:

Значение типа «дата» — окончание расчетного периода журнала расчетов.

Описание:

Метод позволяет определить дату окончания того периода журнала расчетов, в который попадает заданная дата. Работа метода не зависит от наличия в журнале расчетов данных за соответствующий расчетный период.

Пример:

См. предыдущий пример.

См. также: НачалоПериодаПоДате

ПериодПоДате

Возвращает период журнала расчетов по дате.

Синтаксис:

ПериодПоДате(<Дата>)

Англоязычный синоним:

PeriodByDate

Параметры:

<Дата>	Значение типа «дата».

Возвращаемое значение:

Период журнала расчетов.

Описание:

Метод ПериодПоДате возвращает период журнала расчетов по дате — т. е. период, в который попадает заданная дата.

Пример:

```
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
Пер = ЖР.ПериодПоДате(РабочаяДата());
```

См. также: НачалоТекущегоПериода, ПериодДействия, ПериодРегистрации, КонецПериодаПоДате

УстановитьТекущийПериод

Устанавливает текущий период журнала расчетов.

Синтаксис:

УстановитьТекущийПериод(<Период>, <Способ>)

Англоязычный синоним:

SetCurrentPeriod

Параметры:

<Период>	Значение типа «период журнала расчетов».
<Способ>	Необязательный параметр, число: 0 — не отрабатывать системные действия, связанные со сменой периода; 1 -отработать системные процедуры по умолчанию (например, отменить рассчитанность записей при откате назад или провести архивацию документов при смене периода «вперед»). В этом режиме метод ведет себя как интерактивная смена периода но без вопросов. Значение параметра по умолчанию 1.

Возвращаемое значение:

Число: 1 — вополнено; 0 — не выполнено.

Описание:

Метод УстановитьТекущийПериод устанавливает текущий период журнала расчетов. Ведет себя так же, как интерактивная смена текущего периода, но без диалоговых окон.

Пример:

```
//Установить следующий период
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
Пер = ЖР.ТекущийПериод();
Пер = Пер.ПрибавитьПериод(1);
Сообщение("Смена периода журнала займет некоторое время...");
ЖР.УстановитьТекущийПериод(Пер);
```

См. также: НачалоТекущегоПериода, ПериодДействия, ПериодРегистрации, КонецПериодаПоДате

ТекущийПериод

Возвращает текущий расчетный период журнала расчетов.

Синтаксис:

ТекущийПериод()

Англоязычный синоним:
`CurrentPeriod`
Возвращаемое значение:
Значение типа «Период журнала расчетов».
Описание:
Метод предназначен для определения текущего расчетного периода журнала расчетов.
Пример:
```
// Модуль формы журнала расчетов
Пер = ТекущийПериод();
// установим границу просмотра записей на два расчетных периода
ГраницаПросмотра(Пер.ПрибавитьПериод(-1));
```

ПолучитьЗапись

Получить из выборки следующую запись журнала расчетов.
Синтаксис:
`ПолучитьЗапись()`
Англоязычный синоним:
`GetRecord`
Возвращаемое значение:
Числовое значение — 1, если очередная запись журнала расчетов найдена, и 0, если очередная запись журнала не найдена или отсутствует (выборка пуста).
Описание:
Метод предназначен для позиционирования очередной записи журнала расчетов в выборке, установленной одним из следующих методов:

- `ВыбратьЗаписи`
- `ВыбратьЗаписиПоОбъекту`
- `ВыбратьПериод`
- `ВыбратьПериодПоОбъекту`
- `ВыбратьЗаписиПоДокументу`

Как правило, применяется в цикле после использования одного из вышеперечисленных методов.
Пример:
```
// Переберем все записи текущего периода журнала расчетов
//по текущему основному средству и просуммируем результат
ЖР = СоздатьОбъект("ЖурналРасчетов.Амортизация");
ЖР.ВыбратьПериодПоОбъекту(Объект, ЖР.НачалоТекущегоПериода());
// Инициализируем переменную
Сумма = 0;
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   Сумма = Сумма + ЖР.Результат;
КонецЦикла;
```

ВыполнитьРасчет

Выполнить расчет текущей записи журнала расчетов.
Синтаксис:
`ВыполнитьРасчет()`
Англоязычный синоним:
`RunCalculate`
Возвращаемое значение:
Числовое значение — 1, если расчет записи выполнен успешно, и 0 — в противном случае (в том числе, при прерывании процедуры расчета пользователем путем нажатия клавиши <Esc>).
Описание:
Выполняет расчет текущей строки журнала расчетов.
Как правило, применяется в цикле, выполняющем перебор записей журнала расчетов при помощи метода `ПолучитьЗапись`. Работает точно так же, как и метод `Рассчитать`, но имеет возвращаемое значение. Применяется в том случае, если при обработке в цикле записей журнала расчетов бывает необходимо прервать выполнение расчета.
Пример:
```
// процедура выполняет расчет по журналу Зарплата
Процедура Выполнить()
   ЖрнЗарплата.ВыбратьПериод{ЖрнЗарплата.НачалоТекущегоПериода());
   Пока ЖрнЗарплата.ПолучитьЗапись() > 0 Цикл
      Если Считать() = 1 Тогда
         Если ЖрнЗарплата.ВыполнитьРасчет()= 0 Тогда
            // прерываем процедуру, если расчет неудачен или
            // пользователь нажал Esc
            Прервать;
         КонецЕсли;
      КонецЕсли;
```

```
   КонецЦикла;
КонецПроцедуры
```

ОписательПериода

Возвращает строку-описатель расчетного периода.

Синтаксис:

`ОписательПериода(<Дата>)`

Англоязычный синоним:

`PeriodDescriptor`

Параметры:

`<Дата>`	Дата периода, описатель которого требуется получить.

Возвращаемое значение:

Строка — описатель расчетного периода.

Описание:

Метод предназначен для получения строкового представления расчетного периода журнала расчетов. Возвращает значение для того расчетного периода, в который попадает параметр `<Дата>`. Тип формируемой строки различается для разной периодичности журнала расчетов. Например, для журналов с месячной периодичностью строка имеет вид "Январь 1997г", в случае квартальной периодичности — "1 Квартал 1997г", а в случае недельной периодичности — "12.02.97 — 18.02.97".

Как правило, применяется при формировании отчетов.

Пример:

```
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
Сообщить("Текущий период — " + ЖЗ.ОписательПериода(РабочаяДата()));
```

См. также: `НачалоПериодаПоДате`, `КонецПериодаПоДате`

ТекущаяЗапись

Возвращает значение текущей записи журнала расчетов.

Синтаксис:

`ТекущаяЗапись()`

Англоязычный синоним:

`CurrentRecord`

Возвращаемое значение:

Значение текущей записи журнала расчетов.

Описание:

Метод возвращает ссылку на текущую запись (позиционированную в данный момент) журнала расчетов.

Возвращенное значение может, например, затем передаваться в качестве параметра методу журнала расчетов `НийтиЗапись`.

Пример:

```
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖЗ.ВыбратьЗаписиПоОбъекту(Сотрудник, ДН, ДО);
Пока ЖЗ.ПолучитьЗапись() = 1 Цикл
   Если ЖЗ.ПлатВедомость.Выбран() = 1 Тогда
      ЗаписьЖР = ЖЗ.ТекущаяЗапись();
   КонецЕсли;
КонецЦикла;
// теперь найдем запись и что-нибудь с ней поделаем
ЖЗ.НайтиЗапись(ЗаписьЖР);
ЖЗ.ПлатВедомость.Проведена = 1;
ЖЗ.ФиксироватьЗапись();
```

См. также: `НайтиЗапись`

НайтиЗапись

Позиционирует в журнале расчетов заданную запись.

Синтаксис:

`НайтиЗапись(<Запись>)`

Англоязычный синоним:

`FindRecord`

Параметры:

`<Запись>`	Запись журнала расчетов.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод позиционирует запись журнала расчетов, переданную в качестве параметра `<Запись>`. Передаваемое в качестве параметра значение, как правило, ранее получается за счет применения метода журнала расчетов `ТекущаяЗапись`. Фактически, данный метод открывает выборку журнала расчетов, заведомо состоящую из одной записи, и позиционирует ее.

Пример:

```
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖЗ.ВыбратьЗаписиПоОбъекту(Сотрудник, ДН, ДО);
Пока ЖЗ.ПолучитьЗапись() = 1 Цикл
   Если ЖЗ.ПлатВедомость.Выбран() = 1 Тогда
      ЗаписьЖР = ЖЗ.ТекущаяЗапись();
   КонецЕсли;
КонецЦикла;
// теперь найдем запись и что-нибудь с ней поделаем
ЖЗ.НайтиЗапись(ЗаписьЖР);
ЖЗ.ПлатВедомость.Проведена = 1;
ЖЗ.ФиксироватьЗапись();
```

См. также: `ТекущаяЗсшись`

ФиксироватьЗапись

Фиксирует текущую запись журнала расчетов, предотвращая редактирование результата ее расчета.

Синтаксис:

```
ФиксироватьЗапись()
```

Англоязычный синоним:

```
LockRecord
```

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод фиксирует текущую запись журнала расчетов. После применения этого метода атрибут записи `Фиксирована` принимает значение 1. Фиксация записи, фактически, означает невозможность отредактировать результат ее расчета любым, в том числе программным, способом.

Пример:

```
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖЗ.ВыбратьЗаписиПоОбъекту(Сотрудник, ДН, ДО);
Пока ЖЗ.ПолучитьЗапись() = 1 Цикл
   Если (ЖЗ.Документ = ТекущийДокумент()) И
        (ЖЗ.ВидРасч = ВидРасчета.ПоТарифу) Тогда
      ЖЗ.ФиксироватьЗапись();
   КонецЕсли;
КонецЦикла;
```

См. также: атрибут `Фиксирована`, метод `ОсвободитьЗапись`

ОсвободитьЗапись

Снимает фиксацию с текущей записи журнала расчетов.

Синтаксис:

```
ОсвободитьЗапись()
```

Англоязычный синоним:

```
ReleaseRecord
```

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод снимает фиксацию с текущей записи журнала расчетов. После при менения этого метода атрибут записи `Фиксирована` принимает значение 0. Действие, обратное фиксации записи, фактически, означает возможность редактирования результата ее расчета.

Пример:

```
Если ЖрнЗарплата.Фиксирована = 1 Тогда
   Если ДатаДок > ЖрнЗарплата.НачалоТекущегоПериода() Тогда
      ОсвободитьЗапись();
   Иначе
      // отказ от расчета!
      // ....
   КонецЕсли;
КоненцЕсли;
```

См. также: `Фиксирована`, `ФиксироватьЗапись`

ВвестиПерерасчет

Вводит перерасчет текущей записи журнала расчетов.

Синтаксис:

```
ВвестиПерерасчет()
```

Англоязычный синоним:

```
InsertRecalc
```

Возвращаемое значение:

Число: 1 — если операция успешно выполнена, и 0 — в противном случае.

Описание:

Метод предназначен для ввода строк-перерасчетов в журнал расчетов. Как правило, используется в модуле документа, но может использоваться и в модуле вида расчета. Метод вводит полную копию текущей записи в журнал расчетов, не заполняя лишь атрибут `Результат`. Впоследствии система распознает «обычные» записи и записи-перерасчеты.

Записи-перерасчеты могут вводиться только для записей одного из прошлых расчетных периодов. Невозможно ввести перерасчет записи, которая в свою очередь является перерасчетом. При этом система позволяет вводить несколько записей-перерасчетов в разных расчетных периодах для одной и той же записи журнала.

Например, в случае если журнал расчетов имеет месячную периодичность, для «мартовской» записи журнала могут быть введены перерасчеты в апреле, мае и т. д.

Для записей, введенных в журнал одним из методов `ВвестиПерерасчет` или `ВвестиПерерасчетНаОсновании`, атрибут `Перерасчет` принимает значение 1.

При расчете записи-перерасчета результат вычисляется с учетом результата первичной (перерассчитываемой) записи. Это значит, что если в журнале расчетов введен перерасчет записи прошлого периода, то результат перерасчета будет рассчитан за вычетом значения `Результат` первичной записи.

Пример:

```
Процедура Перерасчет(Группа, Сотрудник, Основание, Начало, Окончание)
   //Группа расчетов "оплата по среднему"
   ОС = ГруппаРасчетов.ОплатаПоСреднему;
   // перерасчет по журналу "зарплата"
   ЖрнЗарплата = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   // дата, начиная с которой нужно перерассчитывать и начисления
   ДатаНачислений = ЖрнЗарплата.КонецПериодаПоДате(Начало) + 1;
   // выделим все записи ЖР по сотруднику
   ЖрнЗарплата.ВыбратьЗаписиПоОбъекту(Сотрудник,
                    ЖрнЗарплата.НачалоПериодаПоДате(Начало),
                    Минимум(Окончание, ЖрнЗарплата.НачалоТекущегоПериода() - 1));
   Сч = 1;
   Пока ЖрнЗарплата.ПолучитьЗапись() > 0 Цикл
      Если ЖрнЗарплата.ВидРасч.ВходитВГруппу(Группа) = 1 Тогда
         Пересчитать = 1;
         Если ((ЖрнЗарплата.ВидРасч.ВходитВГруппу(ОС) = 1)И
                (ЖрнЗарплата.ДатаНачала<ДатаНачислений)) Тогда
            // надо перерассчитывать начисления, оплачиваемые
            //по среднему заработку только следующего
            // (за началом действия данного документа-основания)
            // периода расчета
            Пересчитать = 0;
         КонецЕсли;
         Если Пересчитать = 1 Тогда
            ЖрнЗарплата.ВвестиПерерасчет(};
         КонецЕсли;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

См. также: `Перерасчет`, `ВвестиПерерасчетНаОсновании`

ВвестиПерерасчетНаОсновании

Вводит перерасчет текущей записи журнала расчетов на основании произвольного документа.

Синтаксис:

`ВвестиПерерасчетНаОсновании(<Документ>)`

Англоязычный синоним:

`InsertRecalcByReason`

Параметры:

`<Документ>`	Документ, на основании которого вводится запись (или записи) в журнал расчетов.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод предназначен для ввода строк-перерасчетов в журнал расчетов на основании произвольного документа. Как правило, используется в модуле документа, но может использоваться и в модуле вида расчета. Метод вводит полную копию текущей записи в журнал расчетов, не заполняя лишь атрибут `Результат` и вводя в качестве документа-основания параметр `<Документ>`.

В отличие от метода `ВвестиПерерасчет` (который вводит записи журнала на основании того документа, в модуле которого используется метод), данный метод вводит записи-перерасчеты, задавая для них произвольный документ-основание.

Записи-перерасчеты могут вводиться только для записей одного из прошлых расчетных периодов. Невозможно ввести перерасчет записи, которая в свою очередь является перерасчетом. При этом система позволяет вводить несколько записей-перерасчетов в разных расчетных периодах для одной и той же записи журнала.

Пример:

```
Процедура ПровестиПерерасчет(Док)
   //Группа расчетов "оплата по среднему"
   ОС = ГруппаРасчетов.ОплатаПоСреднему;
   // перерасчет по журналу "зарплата"
   ЖрнЗарплата = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   // дата, начиная с которой нужно перерасчитывать и начисления
   ДатаНачислений = ЖрнЗарплата.КонецПериодаПоДате(Начало) + 1;
   // выделим все записи ЖР по сотруднику
   ЖрнЗарплата.ВыбратьЗаписиПоОбъекту(Док.Сотрудник,
               ЖрнЗарплата.НачалоПериодаПоДате(Док.Начало),
               Минимум(Док.Окончание, ЖрнЗарплата.НачалоТекущегоПериода() - 1));
   Сч = 1;
   Пока ЖрнЗарплата.ПолучитьЗапись() > 0 Цикл
      Если ((ЖрнЗарплата.ВидРасч.ВходитВГруппу(ОС) = 1) И
            (ЖрнЗарплата.ДатаНачала < ДатаНачислений)) Тогда
         ЖрнЗарплата.ВвестиПерерасчетНаОсновании(Док);
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

См. также: Перерасчет, ВвестиПерерасчет

Вид

Название журнала расчетов.

Синтаксис:

```
Вид()
```

Англоязычный синоним:

```
Kind
```

Возвращаемое значение:

Строка — название журнала расчетов, как оно задано при конфигурировании.

Описание:

Метод возвращает название журнала расчетов, как оно задано при конфигурировании журнала. Один из вариантов использования метода — определение вида журнала в предопределенной процедуре глобального модуля ПриСменеРасчетногоПериода.

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
```

ПредставлениеВида

Определить пользовательское представление вида журнала расчетов.

Синтаксис:

```
ПредставлениеВида()
```

Англоязычный синоним:

```
KindPresent
```

Возвращаемое значение:

Строковое значение, содержащее пользовательское представление вида журнала расчетов.

Описание:

Метод ПредставлениеВида позволяет получить пользовательское представление вида журнала расчетов, как оно задано в конфигураторе.

Пример:

```
// отобразим пользовательское представление в строке состояния
Состояние(ЖР.ПредставлениеВида());
```

НазначитьТип

Назначить тип для реквизита неопределенного вида.

Синтаксис:

НазначитьТип(<ИмяРеквизита>, <ИмяТипа>, <Длина>, <Точность>}

Англоязычный синоним:

SetType

Параметры:

<ИмяРеквизита>	Строковое выражение — название реквизита журнала расчетов неопределенного типа, как он назван в конфигураторе.
<ИмяТипа>	Строковое выражение — название типа данных (или Вид субконто), который назначается реквизиту журнала расчетов. 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:

```
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
// будем записывать несколько реквизитов
ЖЗ.НазначитьТип("ТМЦ", "Справочник.Товары");
ЖЗ.УстановитьРеквизит("ТМЦ", ВыбТовар);
ЖЗ.ВвестиРасчет(Сотр, ВР, Нач, Оконч, Сумма * Процент);
```

УстановитьРеквизит

Установить значение реквизита журнала расчетов для последующей записи.

Синтаксис:

УстановитьРеквизит(<ИмяРеквизита>, <Значение>)

Англоязычный синоним:

SetAttribute

Параметры:

<ИмяРеквизита>	Строка — наименование реквизита журнала расчетов.
<Значение>	Значение, устанавливаемое для записи в реквизит <ИмяРеквизита>.

Описание:

Метод предназначен для организации заполнения дополнительных реквизитов журнала расчетов при вводе новых записей в журнал расчетов. Метод применяется в двух случаях: во-первых, при вводе записей журнала расчетов в процедуре проведения документа одним из перечисленных ниже методов:

- ВвестиРасчет;
- ВвестиРасчетНаОсновании;
- ЗаписатьРасчет;
- ЗаписатьРасчетНаОсновании.

Во-вторых, при вводе произвольных новых записей журнала расчетов (методы Новая и Записать).

В этих двух случаях метод УстановитьРеквизит работает по-разному. В случае, когда метод применяется перед вводом записей при проведении документа, не имеет смысла устанавливать значения для следующих реквизитов:

- Объект,
- Документ,
- РодительскийДокумент,
- ВидРасчета,
- ДатаНачала,
- ДатаОкончания,
- ПервичнаяЗапись.

Данные установки игнорируются последующими методами ВвестиРасчет, ЗаписатьРасчет и им подобными. Т. е. при вводе записей журнала этими методами возможна установка только реквизитов журнала, заданных при конфигурировании, реквизита Результат и таких признаков записи журнала, как: Рассчитана, Исправлена, Сторно, Фиксирована, Перерасчет.

В случае, когда метод применяется при вводе записей методами Новая и Записать, в качестве первого параметра <ИмяРеквизита> могут применятся наименования реквититов журнала расчетов, заданные при конфигураировании системы или следующие атрибуты, соответствующие предопределенным реквизитам журнала расчетов:

- Документ
- РодительскийДокумент
- Объект
- ВидРасч
- ДатаНачала
- ДатаОкончания
- Сторно

- Рассчитана
- Исправлена
- Фиксирована
- Перерасчет
- ПервичнаяЗапись
- Результат

Внимание! Атрибуты журнала расчетов ПериодДействия и ПериодРегистрации не устанавливаются пользователем. Атрибут ПериодРегистрации при вводе записи принимает значение текущего (установленного в данный момент для журнала расчетов) периода, а атрибут ПериодДействия соответствует установленным значениям ДатаНачала и ДатаОкончания.

Замечание. Установленные методом значения используются однократно, при первом же вызове одного из вышеперечисленных методов ввода новых записей журнала расчетов установленные значения «сбрасываются». Для записи нескольких реквизитов необходимо использовать метод несколько раз.

Пример:

```
Процедура ВвестиЗапись(Сотр, ВР, Сумма, Процент)
   ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   // будем записывать несколько реквизитов
   ЖЗ.УстановитьРеквизит("Сторно", 1);
   ЖЗ.УстановитьРеквизит("Рассчитана", 1);
   // введенные записи будут сторнированными и отмеченными
   // как рассчитанные
   ЖЗ.ВвестиРасчет(Сотр, ВР, Нач, Оконч, Сумма*Процент);
КонецПроцедуры
```

См. также: ВвестиРасчет, ЗаписатьРасчет

ВвестиРасчет

Ввести запись в журнал расчетов.

Синтаксис:

ВвестиРасчет(<Объект>, <ВидРасчета>, <ДатаНачала> , <ДатаОкончания>, <Результат>}

Англоязычный синоним:

InsertCalculation

Параметры:

<Объект>	Объект расчета — элемент справочника, заданного при конфигурировании журнала расчетов.
<ВидРасчета>	Вид вводимого расчета — ссылка на агрегатный объект «ВидРасчета».
<ДатаНачала>	Необязательный параметр. Дата начала действия вводимого расчета. По умолчанию — дата начала текущего периода журнала расчетов.
<ДатаОкончания>	Необязательный параметр. Дата окончания действия вводимого расчета. По умолчанию — дата окончания текущего периода журнала расчетов.
<Результат>	Необязательный параметр. Результат расчета. По умолчанию — 0.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод ВвестиРасчет предназначен для ввода строк в журнал расчетов. Как правило, используется в модуле расчета документа, но может использоваться и в модуле вида расчета. Метод вводит одну или несколько записей в журнал расчетов, и результат ее действия зависит от того, вводится «вытесняющий» расчет или обычный.

При вводе вытесняющего расчета проводится анализ уже существующих записей журнала расчетш; для интервала времени, заданного параметрами ДатаНачала и ДатаОкончания. Если в этом периоде уже существуют записи, приоритет вытеснения которых меньше либо равен приоритету вытеснения вводимого расчета, будет проводиться редактирование периода их действия или полное их удаление. Это приведет, например, к тому, что вводимая запись будет вытеснять записи с тем же видом расчета.

В том случае, если вытесняемые данные принадлежат не текущему периоду расчета, проводится не корректировка или полное удаление записей, а их сторнирование

Если приоритет вытеснения уже существующих записей больше приоритета вытеснения вводимого расчета, то корректируется период действия вводимой записи. Это может привести, например, к тому, что новая запись вовсе не будет введена, если ее место во времени уже занято записями с большим приоритетом вытеснения.

Для случая ввода невытесняющих расчетов анализ существующих записей журнала расчетов и их протяженности во времени не проводится, и расчет вводится в любом случае.

Во всех вышеописанных случаях ввод записей в журнал расчетов проводится в пределах границ расчетных периодов журнала расчетов. Например, если при месячной периодичности журнала задать дату начала и дату окончания, отстоящие друг от друга более, чем на месяц, будет введено более одной записи журнала расчетов — для всех периодов, затрагиваемых временным интервалом от <ДатаНачала> до <ДатаОкончания>.

Важно! Следует помнить, что ввод записи с временем действия расчета, не принадлежащим текущему расчетному периоду, не означает ввода записей прошлого или будущего периода. Вводимые записи всегда принадлежат текущему периоду (т. е. введены в текущем периоде), хотя и производят расчеты прошлого или будущего периодов.

Атрибуты `Документ` и `РодительскийДокумент` журнала расчетов заполняются значением того документа, в модуле расчета которого сработал метод `ВвестиРасчет`. Этот документ будет «родителем» и «основанием» вводимой записи журнала расчетов. В том случае, когда метод сработал в модуле вида расчета, реквизиты `Документ` и `РодительскийДокумент` заполняются значениями этих атрибутов той записи журнала расчетов, при расчете которой выполнен метод `ВвестиРасчет`.

Таким образом, конкретный документ является «родителем» не только тех записей журнала расчетов, которые введены непосредственно им, но и тех записей, которые порождены при расчете записей, введенных документом. Можно считать, что для таких записей документ является не «родителем», а «прародителем».

Параметр `<Результат>` можно использовать в том случае, когда сразу при вводе записей журнала расчетов можно записать их результат.

Важно! Метод `ВвестиРасчет` можно применять только для тех объектов, которые созданы функцией `СоздатьОбъект`. Это значит, например, что его нельзя использовать непосредственно (без точки) в форме журнала расчетов или в модуле видов расчета.

Пример:

```
// Ввод выбранного расчета.
// Атрибут "Расчет" — реквизит формы документа
Процедура ПровестиДокумент()
   // Процедура выполняется при проведении документа расчета
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   ЖР.ВвестиРасчет(Сотрудник, Расчет, ДатаН, ДатаК, 0);
КонецПроцедуры
```

См. также: `ВвестиРасчетНаОсновании`, `ЗаписатьРасчет`

ВвестиРасчетНаОсновании

Ввести запись в журнал расчетов на основании произвольного документа.

Синтаксис:

```
ВвестиРасчетНаОсновании(<Основание>, <Объект>, <ВидРасчета>, <ДатаНачала>,
                <ДатаОконч>, <Результат>)
```

Англоязычный синоним:

`InsertCalculationByReason`

Параметры:

`<Основание>`	Документ, на основании которого вводится запись (или записи) в журнал расчетов.
`<Объект>`	Объект расчета — элемент справочника, заданного при конфигурировании журнала расчетов.
`<ВидРасчета>`	Вид вводимого расчета — ссылка на агрегатный объект типа «ВидРасчета».
`<ДатаНачала>`	Необязательный параметр. Дата начала действия вводимого расчета. По умолчанию — дата начала текущего расчетного периода журнала расчетов.
`<ДатаОконч>`	Необязательный параметр. Дата окончания действия вводимого расчета. По умолчанию — дата окончания текущего расчетного периода журнала расчетов.
`<Результат>`	Необязательный параметр. Результат расчета. По умолчанию — ноль.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод предназначен для ввода строк в журнал расчетов. Работает так же, как и метод `ВвестиРасчет`, но при этом реквизит Документ вводимых строк журнала расчетов заполняется значением параметра `<Основание>`. Использование метода `ВвестиРасчетНаОсновании` — один из способов непосредственно установить родительские отношения между любым документом и вводимыми записями журнала расчетов.

Важно! Метод `ВвестиРасчетНаОсновании` можно применять только для тех объектов, которые созданы функцией `СоздатьОбъект`. Это значит, например, что его нельзя использовать непосредственно (без точки) в форме журнала расчетов или в модуле видов расчета.

Пример:

```
// Модуль выполняется при проведении документа расчета "Начало месяца"
Процедура ПровестиДокумент()
   // Если можно проводить...
   Если (МожноПроводить = 1) Тогда
      // Журнал расчетов — Зарплата
      ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
```

```
        ЖРКомп = СоздатьОбъект("ЖурналРасчетов.Компенсации");
        // конец и начало текущего периода
        КТП = ЖР.КонецТекущегоПериода();
        НТП = ЖР.НачалоТекущегоПериода();
        Сотр = СоздатьОбъект("Справочник.Сотрудники");
        Спр = СоздатьОбъект("Справочник.ПриказыДлительногоДействия");
        Сотр.ВыбратьЭлементы();
        Пока Сотр.ПолучитьЭлемент() > 0 Цикл
           // расчет "приказов длительного действия"
           // типа доплат исп. листов, штрафов и пр...
           Спр.ИспользоватьВладельца(Сотрудник);
           Спр.ВыбратьЭлементы();
           // предполагается, что в каждом приказе есть
           // реквизиты Начало, Окончание и Сотрудник
           Пока Спр.ПолучитьЭлемент() > 0 Цикл
              //. . .
              Если Спр.Приказ.Вид() <> "" Тогда
                 Если ((Спр.Приказ.Окончание >= НТП) И
                       (Спр.Приказ.Начало <= КТП)) Тогда
                    // предполагается, что в каждом приказе
                    // есть реквизиты Начало и Окончание
                    ЖР.ВвестиРасчетНаОсновании(Спр.Приказ, Спр.Приказ.Сотрудник,
                          Спр.Приказ.Расчет, Макс(Спр.Прика з.Начало, НТП),
                          Мин(Спр.Приказ.Окончание, КТП), 0);
                 КонецЕсли;
              КонецЕсли;
           КонецЦикла;
        КонецЦикла;
     ИначеЕсли МожноПроводить = 0 Тогда
        Сообщить("Документ не проведен!!!");
     КонецЕсли;
КонецПроцедуры
```

См. также: ВвестиРасчет, ЗаписатьРасчетНаОсновании

ЗаписатьРасчет

Записать расчет в журнал расчетов.

Синтаксис:

```
ЗаписатьРасчет (<Объект>, <ВидРасчета>, <ДатаНачала>, <ДатаОкончания>,
                <Результат>)
```

Англоязычный синоним:

```
EnterCalculation
```

Параметры:

<Объект>	Объект расчета — элемент справочника, заданного при конфигурировании журнала расчетов.
<ВидРасчета>	Вид вводимого расчета — ссылка на агрегатный объект «Вид Расчета».
<ДатаНачала>	Необязательный параметр. Дата начала действия вводимого расчета. По умолчанию — дата начала текущего периода журнала расчетов.
<датаОкончания>	Необязательный параметр. Дата окончания действия вводимого расчета. По умолчанию — дата окончания текущего периода журнала расчетов.
<Результат>	Необязательный параметр. Результат расчета. По умолчанию — ноль.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод предназначен для ввода строк в журнал расчетов так же, как и метод ВвестиРасчет. Отличие заключается в том, что ввод вытесняющих расчетов приводит к вытеснению только тех расчетов, которые имеют меньший приоритет, а не меньший либо равный, как в случае с методом ВвестиРасчет. Это приводит, в частности, к тому, что за счет применения этого метода расчет не вытесняет «сам себя».

При записи невытесняющего расчета ввод новых записей также происходит «осмотрительно» — новые записи вводятся только в том случае, если в журнале расчетов нет точно такой же записи. Под точно такой же записью здесь подразумевается запись с таким же видом расчета, для того же объекта и с тем же периодом действия.

Важно! Метод ЗаписатьРасчет можно применять только для тех объектов, которые созданы функцией СоздатьОбъект. Это значит, например, что его нельзя использовать непосредственно (без точки) в форме журнала расчетов или в модуле видов расчета.

Пример:

См. предыдущий пример.

См. также: ВвестиРасчет, ЗаписатьРасчетНаОсновании

ЗаписатьРасчетНаОсновании

Ввести запись в журнал расчетов на основании произвольного документа.

Синтаксис:

```
ЗаписатьРасчетНаОсновании(<Основание>, <Объект>, <ВидРасчета>, <ДатаНачала>,
                          <ДатаОконч>, <Результат>)
```

Англоязычный синоним:

```
EnterCalculationByReason
```

Параметры:

<Основание>	Документ, на основании которого вводится запись (или записи) в журнал расчетов.
<Объект>	Объект расчета — элемент справочника, заданного при конфигурировании журнала расчетов.
<ВидРасчета>	Вид вводимого расчета — ссылка на агрегатный объект «Вид Расчета».
<ДатаНачала>	Необязательный параметр. Дата начала действия вводимого расчета. По умолчанию — дата начала текущего периода журнала расчетов.
<ДатаОконч>	Необязательный параметр. Дата окончания действия вводимого расчета. По умолчанию — дата окончания текущего периода журнала расчетов.
<Результат>	Необязательный параметр. Результат расчета. По умолчанию — ноль.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод предназначен для ввода строк в журнал расчетов. Работает так же, как и метод ЗаписатьРасчет, но при этом реквизит Документ вводимых строк журнала расчетов заполняется значением параметра <Основание>. Использование метода ЗаписатьРасчетНаОсновании — один из способов непосредственно установить родительские отношения между любым документом и вводимыми записями журнала расчетов.

Данный метод отличается от метода ВвестиРасчетНаОсновании тем, что ввод вытесняющих расчетов приводит к вытеснению только тех расчетов, которые имеют меньший приоритет, а не меньший либо равный, как в случае с методом ВвестиРасчетНаОсновании. Это приводит, в частности, к тому, что за счет применения этого метода расчет не вытесняет «сам себя».

При записи невытесняющего расчета ввод новых записей также происходит «осмотрительно» — новые записи вводятся только в том случае, если в журнале расчетов нет точно такой же записи. Под точно такой же записью здесь подразумевается запись с таким же видом расчета, для того же объекта и с тем же периодом действия.

Важно! Метод ЗаписатьРасчетНаОсновании можно применять только для тех объектов, которые созданы функцией СоздатьОбъект. Это значит, например, что его нельзя использовать непосредственно (без точки) в форме журнала расчетов или в модуле видов расчета.

Пример:

```
Процедура Перерасчет(Группа, Сотрудник, Основание, Начало, Окончание) Экспорт
   // перерасчет по журналу зарплата
   ЖР = СоздатьОбъект{"ЖурналРасчетов.Зарплата");
   ЖР.ВыбратьЗаписиПоОбъекту(Сотрудник, ЖР.НачалоПериодаПоДате(Начало),
       Мин(Окончание, ЖР.НачалоТекущегоПериода() - 1));
   Сч = 1;
   Пока (Сч < 50...) Цикл
      Если (Расчеты[Сч].ВходитВГруппу(ВсеУд)=1) Тогда
         ЖР.ЗаписатьРасчетНаОсновании(Основание, Сотрудник, Расчеты[Сч],
                  Начала[Сч], Окончания[Сч], 0);
      Иначе
         ЖР.ВвестиРасчетНаОсновании(Основания[Сч], Сотрудник, Расчеты[Сч],
                  Начала[Сч], Окончания[Сч], 0);
      КонецЕсли;
      Сч = Сч + 1;
   КонецЦикла;
КонецПроцедуры
```

См. также: ВвестиРасчетНаОсновании

Рассчитать

Провести расчет текущей записи.

Синтаксис:

```
Рассчитать()
```

Англоязычный синоним:

```
Calculate
```

Описание:

Выполняет расчет текущей строки журнала расчетов. Фактически, выполняет модуль вида расчета текущей строки журнала расчетов, точнее — предопределенную процедуру этого модуля ПровестиРасчет. Как правило, применя-

ется в цикле, выполняющем перебор записей журнала расчетов при помощи метода ПолучитьЗапись. Расчет записей прошлых расчетных периодов не производится.

При расчете фиксированных записей текущего расчетного периода, модуль вида расчета выполняется, но изменение результата расчета не производится.

Метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
// Рассчитаем все основные средства категории определенного типа
ЖР = СоздатьОбъект("ЖурналРасчетов.Амортизация");
ЖР.ВыбратьПериод(ЖР.НачалоТекущегоПериода());
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   Если ЖР.Объект.Тип = Тип;
      ЖР.Рассчитать();
   КонецЕсли;
КонецЦикла;
```

См. также: ВыполнитьРасчет

ВыбратьЗаписи

Выбрать записи, действующие в определенном временном интервале.

Синтаксис:

ВыбратьЗаписи(<Начало>, <Окончание>}

Англоязычный синоним:

SelectRecords

Параметры:

<Начало>	Дата начала периода.
<Окончание>	Дата окончания периода.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена и выборка не пуста, т. е. содержит хотя бы одну запись; 0 — в противном случае.

Описание:

Метод открывает выборку записей журнала расчетов. Выбираются все записи, период действия которых хоть на один день затрагивается тем периодом, который задан параметрами <Начало> и <Окончание>. Это значит, например, что если запись журнала расчетов имеет даты начала и окончания '15.12.96' и '25.12.96' то она попадет в выборку, открытую при помощи любого из следующих методов:

- ВыбратьЗаписи(•10.10.96', •15.12.96')
- ВыбратьЗаписи('25.12.96', '01.01.97')
- ВыбратьЗаписи('17.12.96', '17.12.96').

Как правило, после применения данного метода проводится перебор всех записей выборки в цикле при помощи метода журнала расчетов ПолучитьЗапись.

Данный метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
// Расчет количества дней, отработанных всеми сотрудниками
// за прошлый период
//
// журнал расчета зарплаты
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
// дата начала прошлого периода
ДатаНачала = ЖР.НачалоПериодаПоДате(ЖР.НачалоТекущегоПериода() - 1);
ЖР.ВыбратьЗаписи(ДатаНачала, ЖР.КонецПериодаПоДате(ДатаНачала));
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   Сумма = Сумма + ЖР.ДНИ;
КонецЦикла;
```

См. также: методы ВыбратьЗаписиПоОбъекту, ВыбратьПериод, атрибуты ДатаНачала, ДатаОкончания

ВыбратьПериод

Выбрать записи, введенные в журнал в определенном расчетном периоде.

Синтаксис:

ВыбратьПериод(<Дата>)

Англоязычный синоним:

SelectPeriod

Параметры:

<Дата>	Дата, лежащая в выбираемом периоде.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена и выборка не пуста, т. е. содержит хотя бы одну запись; 0 — в противном случае.

Описание:

Метод открывает выборку записей журнала расчетов. Выбираются все записи за тот расчетный период, в который попадает <Дата>. Выбираются именно те записи, которые введены в заданном периоде, но не записи, имеющие дату начала и дату окончания, лежащие в заданном периоде. Следовательно, метод ВыбратьПериод отличается от метода ВыбратьЗаписи лишь тем, что осуществляет выборку в другом временном разрезе. В первом случае — это выбор записей по времени их появления (регистрации) в системе, т. е. по расчетным периодам, во втором случае — это выбор записей по времени их действия. Как правило, применяется перед циклом, выполняющим перебор записей журнала расчетов при помощи метода ПолучитьЗапись.

Данный метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
// Расчет суммы амортизации по результатам всех расчетов
// в текущем периоде
// журнал расчета амортизации
ЖР = СоздатьОбъект("ЖурналРасчетов.Амортизация");
ЖР.ВыбратьПериод(ЖР.НачалоТекущегоПериода());
Пока ЖР.ПолучитьЗапись()>0 Цикл
   Сумма = Сумма + ЖР.Результат;
КонецЦикла;
```

См. также: методы ВыбратьПериодПоОбъекту, ВыбратьЗаписи, атрибут ПериодРегистрации

ВыбратьЗаписиПоОбъекту

Выбрать записи, действующие в определенном временном интервале и принадлежащие одному объекту расчета.

Синтаксис:

ВыбратьЗаписиПоОбъекту(<Объект>, <Начало>, <Окончание>)

Англоязычный синоним:

SelectRecordsByObject

Параметры:

<Объект>	Объект расчета, записи которого выбираются.
<Начало>	Дата начала периода.
<Окончание>	Дата окончания периода.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена и выборка не пуста, т. е. содержит хотя бы одну запись; 0 — в противном случае.

Описание:

Метод открывает выборку записей журнала расчетов. Выбираются все записи, рассчитанные для объекта, заданного параметром <Объект>, период действия которых хоть на один день затрагивается тем периодом, который задан параметрами <Начало> и <Окончание>. Следовательно, метод ВыбратьЗаписиПоОбъекту отличается от ВыбратьЗаписи лишь тем, что в первом случае в выборку попадают записи по одному конкретному объекту расчета, а во втором — по всем объектам расчета.

Данный метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
// Проходя по справочнику акционеров, проведем по некоторому
// условию полный расчет дивидендов
// Готовим справочник акционеров
Акционеры = СоздатьОбъект("Справочник.Акционеры");
Акционеры.ПорядокКодов();
Акционеры.ВыбратьЭлементы();
// Журнал расчетов дивидендов
ЖР = СоздатьОбъект("ЖурналРасчетов.Дивиденды");
// Начало и конец текущего периода
Нач = ЖР.НачалоТекущегоПериода();
Кон = ЖР.КонецТекущегоПериода();
Пока Акционеры.ПолучитьЭлемент() > 0 Цикл
   Если Акционеры.Статус = 2 Тогда
      ЖР.ВыбратьЗаписиПоОбъекту(Акционеры.ТекущийЭлемент(), , );
      Пока ЖР.ПолучитьЗапись() > 0 Цикл
         ЖР.Рассчитать();
      КонецЦикла;
   КонецЕсли;
КонецЦикла;
```

См. также: ВыбратьПериодПоОбъекту, ВыбратьЗаписи

ВыбратьЗаписиПоДокументу

Выбрать записи по документу-основанию.

Синтаксис:

ВыбратьЗаписиПоДокументу(<Документ>)

Англоязычный синоним:

SeleetRecordsByDocument

Параметры:

<Документ>	Документ расчета, который является документом-основанием для отбираемых записей.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена и выборка не пуста, т. е. содержит хотя бы одну запись; 0 — в противном случае.

Описание:

Метод открывает выборку записей журнала расчетов. Выбираются все записи, документом-основанием для которых служит заданный <Документ>, вне зависимости от того, в каком расчетном периоде записи порождались. Как и все остальные методы выборки журнала расчетов, как правило, применяется перед циклом, в котором перебираются записи. Работает только для переменных, созданных функцией СоздатьОбъект.

Важно! Документом-основанием записи журнала расчетов может являться:

- документ, непосредственно породивший записи в систему за счет применения методов ВвестиРасчет и ЗаписатьРасчет;
- документ, ссылка на который передана в качестве параметра одному из методов: ВвестиРасчетНаОсновании или ЗаписатьРасчетНаОсновании.

Пример:

```
// Рассчитаем все записи по наряду (процедура выполняется
// в контексте журнала расчетов)
// Журнал расчетов
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьЗаписиПоДокументу(Документ);
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   ЖР.Рас считать();
КонецЦикла;
```

См. также: методы ВыбратьПериод, ВыбратьtЗаписи, атрибут Документ

ВыбратьПериодПоОбъекту

Выбрать записи по одному объекту расчета, зарегистрированные в определенном расчетном периоде журнала расчетов.

Синтаксис:

```
ВыбратьПериодПоОбъекту(<Объект>, <Дата>)
```

Англоязычный синоним:

```
SelectPeriodByObject
```

Параметры:

<Объект>	Объект расчета, записи которого выбираются.
<Дата>	Дата, лежащая в выбираемом периоде.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена и выборка не пуста, т. е. содержит хотя бы одну запись; 0 — в противном случае.

Описание:

Метод открывает выборку записей журнала расчетов. Работает так же, как метод ВыбратьПериод, за исключением того, что выбираются все записи для того объекта расчета, который задан параметром <Объект>.

Данный метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
*
Процедура Начислено(Сотрудник)
   Начисл = 0;
   ПН = 0;
   ЖрнЗарплата.ВыбратьПериодПоОбъекту (Сотрудник, ДатаАктуальности);
   Пока ЖрнЗарплата.ПолучитьЗапись() = 1 Цикл
      Если ЖрнЗарплата.ВидРасч.ВходитВГруппу(ГруппаРасчетов.ВсеНачисления)-1 Тогда
         Начисл = Рез + ЖрнЗарплата.Результат;
      ИначеЕсли ЖрнЗарплата.ВидРасч = ВидРасчета.ПодоходныйНалог
         Тогда ПН = ПН+ЖрнЗарплата.Результат;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
*
// расчет начального сальдо
ЖрнЗарплата.ВыбратьПериодПоОбъекту(Сотрудник, НТП - 1);
Сальдо = 0;
Пока ЖрнЗарплата.ПолучитьЗапись() = 1 Цикл
   Если ЖрнЗарплата.ВидРасч.ВходитВГруппу(ГруппаРасчетов.ВсеУдержания) = 1 Тогда
      Сальдо = Сальдо-ЖрнЗарплата.Результат;
      // выплаты с деполнента и выплаты дивидендов — не считатем!
```

```
   ИначеЕсли ВыплатаЗрп(ЖрнЗарплата.ВидРасч, ЖрнЗарплата.Документ) = 1 Тогда
      Сальдо = Сальдо - ЖрнЗарплата.Результат;
   ИначеЕсли ЖрнЗарплата.ВидРасч.
                  ВходитВГруппу(ГруппаРасчетов.ПоложительноеСальдо) = 1 Тогда
      Сальдо = Сальдо + ЖрнЗарплата.Результат;
   КонецЕсли;
КонецЦикла;
Если Сальдо <> 0 Тогда
   ЖрнЗарплата.ЗаписатьРасчет(Сотрудник,
   ВидРасчета.НачальноеСальдо, НТП, КТП, Сальдо);
КонецЕсли;
```

См. также: методы ВыбратьПериод, ВыбратьтЗаписи, атрибут Объект

ВыбратьПоЗначению

Осуществляет выборку записей по значению в графе отбора.

Синтаксис:

```
ВыбратьПоЗначению(<ИмяГрафы>, <ЗначениеОтбора>, <ПериодНачала>,
                  <ПериодОкончания>)
```

Англоязычный синоним:

```
SelectByValue
```

Параметры:

<ИмяГрафы>	Строковое значение. Наименование графы отбора, как она определена при конфигурировании.
<ЗначениеОтбора>	Отбираемое значение.
<ПериодНачала>	Значение типа «период журнала расчетов» -первый расчетный период отбора.
<ПериодОкончания>	Значение типа «период журнала расчетов» -последний расчетный период отбора.

Возвращаемое значение:

Число: 1 — если отбор успешно установлен; 0 — в противном случае.

Описание:

Метод предназначен для выборки записей журнала расчетов, которые содержат в графе отбора <ИмяГрафы> заданное <ЗначеяиеОтбора>, период регистрации которых лежит в пределах, заданных параметрами <ПериодНачала> и <ПериодОкончания>.

Если первый параметр задан неверно — т. е. нет такой графы отбора, выборка не будет выполнена и метод возвратит 0. Если заданное <ЗначениеОтбора> не содержится ни в одной строке журнала расчетов, выборка окажется пустой (т. е. в нее не попадет ни одна запись журнала расчетов), но метод возвратит значение 1.

Метод используется так же, как и другие методы создающие выборки записей в журнале расчетов, такие как ВыбратьЗаписи, ВыбратьПериод и т. д. Как и все остальные методы выборки журнала расчетов, как правило, применяется перед циклом, в котором перебираются записи. Работает только для переменных, созданных функцией СоздатьОбъект.

Замечание. Наименованиями граф отбора журнала расчетов могут служить реквизиты справочника объектов расчета, заданного для журнала расчетов, а также значения «Владелец» и «Родитель» этого справочника. Это значит, что в общем случае, в качестве первого параметра, методу могут передаваться предопределенные имена «Родитель» и «Владелец», а также названия реквизитов справочника.

Какие именно графы отбора задействованы в журнале расчетов, задается при конфигурировании.

Данный метод работает так же как и метод формы журнала расчетов УстановитьОтбор, но, разумеется не осуществляет визуального управления формой журнала.

Пример:

```
// Рассчитаем все записи текщего расчетного периода по
// подразделению (переменная Подр)
Процедура РасчетПодразделения(Подр)
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   ТекПер = ЖР.ТекущийПериод();
   ЖР.ВыбратьПоЗначению("Подразделение", Подр, ТекПер.ТекПер);
   Пока ЖР.ПолучитьЗапись() > 0 Цикл
      ЖР.Рассчитать();
   КонецЦикла;
КонецПроцедуры
```

См. также: УстановитьОтбор

Новая

Создает новую запись журнала расчетов.

Синтаксис:

```
Новая()
```

Англоязычный синоним:

```
New
```

Описание:

Метод инициализирует создание новой строки журнала расчетов. Собственно запись новой строки журнала расчетов происходит при вызове метода Записать. После инициализации создания новой строки, как правило, производится заполнение ее реквизитов с последующим вызовом метода Записать.

Пример:

```
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.Новая();
ЖР.Записать();
```

Записать

Вносит изменения записи в журнал расчетов.

Синтаксис:

```
Записать()
```

Англоязычный синоним:

```
Write
```

Описание:

Метод Записать вносит изменения записи или новую запись в журнал расчетов. Данный метод применяется после метода Новая и заполнения реквизитов журнала расчетов при помощи метода УстановитьРеквизит.

Данный метод проверяет корректность заполненных реквизитов журнала расчетов. При вводе новых записей журнала расчетов методами Новая и Записать обязательно должны быть заполнены следующие реквизиты записи журнала: Объект, Документ, ВидРасч. Кроме того, проверяется корректность реквизитов ДатаНачала и ДатаОкончания. Если реквизиты ДатаНачала и ДатаОкончания не установленвы явным образом, при записи они устанавливаются как начало и окончание текущего расчетного прериода, соответственно. Реквизит ПериодРегистрации заполняется текущим значением расчетного периода, установленным для журнала расчетов (см. метод УстановитьПериодРасчета). Если реквизит РодительскийДокумент не установлен явным образом, для него устанавливается то же значение, что и для реквизита Документ. Если реквизиты Рассчитана, Сторно, Перерасчет, Фиксирована не установлены, то запись вводится как простая нерассчитанная, нефиксированная запись.

Внимание! При вводе новых записей в журнал расчетов методами Новая и Записать записи вводятся «как есть». Система не выполняет правила перерасчетов, а также правила взаимного вытеснения видов расчета. Ввод произвольных записей журнала расчетов очень ответственная операция. При использовании этих методов следует внимательно следить за логической целостностью журнала расчетов.

Пример:

```
Перем Док;
Перем Сотр;
Перем Рез;
// документы
Док = СоздатьОбъект("Документ");
// ...позиционируется нужный документ
// сотрудники
Сотр = СоздатьОбъект("Справочник.Сотрудники");
// ...позиционируется нужный элемент справочника сотрудники
// считаем результат...
Рез =
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.Новая();
ЖР.УстановитьПериод(ЖР.ПолучитьпериордПоДате(ТекДата));
ЖР.УстановитьРеквизит("Документ", Док.ТекущийДокумент());
ЖР.УстановитьРеквизит("Объект", Сотр.ТекущийЭлемент());
ЖР.УстановитьРеквизит("ВидРасч", Вид Расчета.ПоОкладу);
ЖР.УстановитьРеквизит("Рассчитана", 1);
ЖР.УстановитьРеквизит("Результат", Рез);
ЖР.Записать();
```

УдалитьЗапись

Удалить запись журнала расчетов.

Синтаксис:

```
УдалитьЗапись()
```

Англоязычный синоним:

```
DeleteRecord
```

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод предназначен для удаления записей журнала расчетов. Данный метод работает только для переменных, созданных функцией СоздатьОбъект.

Пример:

```
// Процедура выполняется в контексте журнала расчетов
// Удалим записи, по которым нет рабочих дней!
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
Календ = СоздатьОбъект("Календарь.Служащие");
ЖР.ВыбратьПериод(ЖР.НачалоТекущегоПериода());
Пока ЖР.ПолучитьЗапись() > 0 Цикл
   Если Календ.Дней(ЖР.ДатаНачала, ЖР.ДатаОкончания) = 0 Тогда
      ЖР.УдалитьЗапись();
   КонецЕсли;
КонецЦикла;
```

Исправить

Исправление результата записи журнала расчетов.

Синтаксис:

`Исправить(<НовыйРезультат>)`

Англоязычный синоним:

`Update`

Параметры:

`<НовыйРезультат>`	Новый результат записи журнала расчетов.

Описание:

Метод выполняет то же действие, что и интерактивная процедура исправления результата, т. е. при исправлении записи текущего расчетного периода просто редактируется результат и метится запись «ручкой», а при исправлении записи прошлого периода — вводится сторно той записи и еще одна запись-исправление. Описание интерактивной процедуры исправления результата можно посмотреть в книге «Руководство пользователя».

ОтменитьИсправление

Отменяет исправление записи журнала расчетов, сделанное ранее.

Синтаксис:

`ОтменитьИсправление()`

Англоязычный синоним:

`UndoUpdate`

Описание:

Отменяет исправление, сделанное ранее (вручную или из языка методом `Исправить`). Метод выполняет те же действие, что и интерактивная процедура отмены исправления записи журнала расчетов. Описание интерактивной процедуры отмены исправления результата записи журнала расчетов можно посмотреть в книге «Руководство пользователя».

Методы контекста Модуля формы журнала расчетов

Описанные в данном разделе методы доступны только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

ВидыОтбора

Установить доступные виды отборов журнала для вызова их в интерактивном режиме.

Синтаксис:

`ВидыОтбора(<СписокИмеяОтборов>)`

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 расчетов (см. «Виды программных модулей»).

Пример:

```
ВидыОтбора("Сотрудник, Расчетчик");
```

УстановитьОтбор

Установить выборку по графе отбора.

Синтаксис:

`УстановитьОтбор(<ИмяГрафыОтбора>, <ЗначениеОтбора>)`

Англоязычный синоним:

SetSelection

Параметры:

<ИмяГрафыОтбора>	Строковое значение. Наименование графы отбора, как она определена при конфигурировании.
<ЗначениеОтбора>	Отбираемое значение.

Возвращаемое значение:

Число: 1 — если отбор успешно установлен; 0 — в противном случае.

Описание:

Метод предназначен для выборки записей журнала расчетов, которые содержат в графе отбора <ИмяГрафыОтбора> заданное <ЗначениеОтбора>. Если первый параметр задан неверно — т. е. нет такой графы отбора, выборка не будет выполнена и метод возвратит 0. Если заданное <ЗначениеОтбора> не будет найдено ни в одной строке журнала расчетов, выборка окажется пустой (т. е. в нее не попадет ни одна запись журнала расчетов), но метод возвратит значение 1.

Если метод сработал, т. е. выборка реально установлена, происходит перерисовка окна журнала расчетов так же, как это происходит при интерактивном выполнении операции отбора.

Для отказа от отбора по значению необходимо вызвать данный метод, указав в качестве первого параметра пустую строку.

Замечание. Наименованиями граф отбора журнала расчетов могут служить реквизиты справочника объектов расчета, заданного для журнала расчетов, а также значения «Владелец» и «Родитель» этого справочника. Это значит, что в общем случае, в качестве первого параметра, методу могут передаваться предопределенные имена «Родитель» и «Владелец», а также названия реквизитов справочника.

Какие именно графы отбора задействованы в журнале расчетов, задается при конфигурировании.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
// модуль формы журнала расчетов
// процедура снимает отбор
Процедура ВыбратьВсе();
   УстановитьОтбор("");
КонецПроцедуры
// Установим отбор сразу при открытии журнала расчетов Зарплата
// Расчетчик — глобальная переменная со ссылкой на группу
// сотрудников
УстановитьОтбор("Родитель", Расчетчик);
```

См. также: ПолучитьОтбор, ЗакладкиОтбора

ПолучитьОтбор

Получить значение предварительно установленного отбора.

Синтаксис:

ПолучитьОтбор(<ИмяГрафыОтбора>, <ЗначениеОтбора>)

Англоязычный синоним:

GetSelection

Параметры:

<ИмяГрафыОтбора>	Переменная в которую будет возвращено строковое значение — наименование графы отбора, как она определена при конфигурировании.
<ЗначениеОтбора>	Переменная, в которую будет возвращено ранее установленное значение отбора.

Возвращаемое значение:

Число: 1 — если отбор был установлен и <ИмяГрафыОтбора> и <ЗначениеОтбора> заполнены установками сделанными ранее методом УстановитьОтбор; 0 — если отбор прежде не был установлен.

Описание:

Метод предназначен для получение текущих значений отбора, установленных в форме журнала расчетов программным образом или интерактивно. Метод заполняет параметры <ИмяГрафыОтбора> и <ЗначениеОтбора>, переданные по ссылке. Если отбор ранее не был установлен, метод возвращает 0, а значения переменных, переданных в качестве параметров, не изменяются.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
// модуль формы журнала расчетов
Перем ИмяОтбора;
Перем ЗначОтб;
Процедура ОтборПодразделения(Подразд)
   // ниже ИмяОтбора и ЗначениеОтбора будут заполнены
   // текущими значениями
   БылОтбор = ПолучитьОтбор(ИмяОтбора, ЗначОтб);
   Если БылОтбор = 1 Тогда
      Если ИмяОтбора = "Родитель" Тогда
         // снимем отбор
```

```
            УстановитьОтбор("");
         ИначеЕсли ИмяОтбора = "Подразделение" Тогда
            Сообщить("Было отобрано " + ЗначОтб.Наименование);
            УстановитьОтбор(ИмяОтбора, Подразд);
         КонецЕсли;
      КонецЕсли;
КонецПроцедуры
```

См. также: УстановитьОтбор, ЗакладкиОтбора

ЗакладкиОтбора

Установить в форме журнала расчетов закладки для интерактивного осуществления отбора.

Синтаксис:

```
ЗакладкиОтбора(<ИмяОтбора>, <ЗначениеОтбора>)
```

Англоязычный синоним:

```
TabCtrlSelection
```

Параметры:

<ИмяОтбора>	Строковое выражение, содержащее имя графы отбора, по которой будут созданы закладки.
<ЗначениеОтбора>	Значение отбора, соответствующее закладке устанавливаемой первоначально.

Возвращаемое значение:

Число: 1 — если операция успешно выполнена; 0 — в противном случае.

Описание:

Метод ЗакладкиОтбора устанавливает в форме журнала расчетов закладки (tab control), позволяющие быстро переключаться между разными значениями по одному и тому же отбору.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
// в форме журнала расчетов "зарплата" установим закладки
// по умолчанию будет использована закладка, соответств.
// текущему расчетчику
ЗакладкиОтбора("Родитель", Расчетчик);
```

См. также: УстановитьОтбор, ПолучитьОтбор

ГраницаПросмотра

Установить в форме журнала расчетов границу видимости записей.

Синтаксис:

```
ГраницаПросмотра(<Период>)
```

Англоязычный синоним:

```
ViewBound
```

Параметры:

<Период>	Значение типа «Период журнала расчетов», которое устанавливается как граница видимости записей журнала расчетов.

Возвращаемое значение:

Значение типа «Период журнала расчетов». Текущее значение границы просмотра записей журнала расчетов, установленное до вызова метода программным образом или интерактивно.

Описание:

Метод ГраницаПросмотра устанавливает в форме журнала расчетов границу видимости записей. Видимыми (т. е. отображаемыми в окне журнала расчетов) становятся записи журнала с периодом регистрации от расчетного периода, заданного параметром <Период> до текущего расчетного периода. После вызова данного метода происходит перерисовка окна, так же как и при интерактивной смене границы просмотра журнала расчетов.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
// в форме журнала расчетов "зарплата" отодвинемся
// на два расчетных периода назад
СтараяГраница = ГраницаПросмотра();
ГраницаПросмотра(Ст.ПрибавитьПериод(-2));
```

См. также: методы РежимПредставления, УстановитьОтбор, атрибут ПериодРегистрации

УстановитьПредставление

Установить в форме журнала расчетов режим представления записей.

Синтаксис:

```
УстановитьПредставление(<Режим>, <Объект>)
```

Англоязычный синоним:

```
SetPresentation
```

Параметры:

<Режим>	Числовое значение — режим представления журнала расчетов.
<Объект>	Объект, записи которого отображаются. Если устанавливаемый режим — «по одному документу» (см. ниже), этот параметр должен иметь тип «документ», если режим устанавливаемого пред-

ставления — «по одному объекту», этот параметр должен иметь тип «элемент справочника».

Возвращаемое значение:

Число: 1, если представление успешно установлено; 0 — если представление по каким-либо причинам не установлено.

Описание:

Метод УстановитьПредставление устанавливает режим вывода записей в форме журнала расчетов. Значение <Режим> может принимать следующие значения:

1. Записи по всем объектам расчета.
2. Записи по одному объекту расчета.
3. Записи по одному документу-основанию.

Параметр <Объект> должен иметь тот или иной тип значения, в зависимости от устанавливаемого режима. Если устанавливается режим просмотра записей по одному документу, то тип значения этого параметра — «документ»; если устанавливается режим просмотра записей по одному объекту расчета (элементу справочника-родителя данного журнала расчетов), то тип значения этого параметра — «элемент справочника».

Если устанавливаемый режим — просмотр записей по всем объектам расчета, параметр <Объект> не используется.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
Перем СтРежим;
Перем СтОбъект;
Перем СменилиРежим;
Процедура ПоОдному(Сотрудник)
   // запомним предыдущие установки
   ПолучитьПредставление(СтРежим, СтОбъект);
   // Сменим режим
   СменилиРежим = УстановитьПредставление(2, Сотрудник);
КонецПроцедуры

Процедура Восстановить()
   Если СменилиРежим = 1 Тогда
      УстановитьПредставление(СтРежим, СтОбъект);
   КонецЕсли;
КонецПроцедуры
```

См. также: ПолучитьПредставление, ГраницаПросмотра

ПолучитьПредставление

Получить установки представления журнала расчетов — режим представления и отображаемый объект.

Синтаксис:

```
ПолучитьПредставление(<Режим>, <Объект>}
```

Англоязычный синоним:

```
GetPresentation
```

Параметры:

<Режим>	Переменная, в которую записывается числовое значение — текущий режим представления журнала расчетов.
<Объект>	Переменная, в которую записывается объект, записи которого отображаются. Если текущий режим — «по одному документу» (см. ниже), этот параметр примет значение типа «документ», если режим устанавливаемого представления — «по одному объекту», этот параметр примет значение типа «элемент справочника».

Возвращаемое значение:

Число: 1, если представление успешно установлено; 0 — если представление по каким-либо причинам не установлено.

Описание:

Метод ПолучитьПредставление считывает текущие установки вывода записей в форме журнала расчетов. Параметр <Режим>, переданный по ссылке, в результате работы метода может принять следующие значения:

1. Записи по всем объектам расчета.
2. Записи по одному объекту расчета.
3. Записи по одному документу-основанию.

Параметр <Объект> получит значение того или иного типа, в зависимости от устанавливаемого режима. Если текущий режим — «по одному документу», этот параметр примет значение типа «документ», если режим устанавливаемого представления — «по одному объекту», этот параметр примет значение типа «элемент справочника».

Если текущий режим представления — просмотр записей по всем объектам расчета, параметр <Объект> не заполняется и его значение остается прежним.

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
Перем СтРежим;
Перем СтОбъект;
Перем СменилиРежим;
```

```
Процедура ПоОдному(Сотрудник)
   // запомним предыдущие установки
   ПолучитьПредставление(СтРежим, СтОбъект);
   // Сменим режим
   СменилиРежим = УстановитьПредставление(2.Сотрудник);
КонецПроцедуры

Процедура Восстановить()
   Если СменилиРежим = 1 Тогда
      УстановитьПредставление(СтРежим, СтОбъект);
   КонецЕсли;
КонецПроцедуры
```

См. также: УстановитьПредставление, ГраницаПросмотра

РассчитыватьПриОтмененИсправления

Устанавливает значение флага автоматического расчета записи журнала расчетов при интерактивной отмене ее исправления.

Синтаксис:

РассчитыватьПриОтмененИсправления(<Флаг>)

Англоязычный синоним:

CalculateOnEditCancel

Параметры:

<Флаг> Необязательный параметр. Число: 1 — автоматически рассчитывать записи журнала расчетов при интерактивной отмене ее исправления; 0 — не рассчитывать автоматически.

Возвращаемое значение:

Текущее числовое значение флага (до испольнения метода).

Описание:

Метод РассчитыватьПриОтмененИсправления устанавливает значение флага автоматического расчета записи журнала расчетов при интерактивной отмене ее исправления. По умолчанию форма журнала расчетов всегда открывается со значением флага 0 (не рассчитывать записи журнала расчетов автоматически).

Данный метод доступен только в контексте Модуля формы журнала расчетов (см. «Виды программных модулей»).

Пример:

```
РассчитыватьПриОтмененИсправления(1);
```

Предопределенные процедуры Модуля формы журнала расчетов

Описанные в данном разделе системные предопределенные процедуры могут располагаться только в программном модуле формы журнала расчетов (см. «Виды программных модулей»).

Данные процедуры предназначены для расширения возможности программного управления правами пользователей на выполнение тех или иных действий.

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».

ПриИсправленииРезультата

Предопределенная процедура при редактировании результата расчета записи журнала расчетов.

Синтаксис:

ПриИсправленииРезультата(<Запись>}

Англоязычный синоним:

OnResuitUpdate

Параметры:

<Запись> Ссылка на запись журнала расчетов, результат расчета которой исправляется.

Описание:

Вызов предопределенной процедуры ПриИсправленииРезультата производится системой 1С:Предприятие неявно при попытке «ручного» исправления результата расчета записи журнала расчетов. Если в данной предопределенной процедуре установить статус возврата — 0, то результат не будет отредактирован.

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриИсправленииРезультата(Запись)
   // группа нередактируемых расчетов
   НГ = ГруппаРасчетов.Нередактируемые;
   Если Запись.ВидРасч.ВходитВГруппу(НГ) = 1 Тогда
      Предупреждение("Операция не выполняется");
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, ПриОтмененИсправления

ПриОтмененИсправления

Предопределенная процедура при отказе от ручного редактирования записи.

Синтаксис:

ПриОтмененИсправления(<Запись>)

Англоязычный синоним:

OnUpdateCancel

Параметры:

<Запись>	Ссылка на запись журнала расчетов, исправление результата которой отменяется.

Описание:

Вызов предопределенной процедуры ПриОтмененИсправления производится системой 1С:Предприятие неявно при попытке отмены «ручного» исправления результата расчета записи журнала расчетов. Это действие вызывается пунктом «Отменить ручное редактирование» меню «Действия» при работе с журналом расчетов. Если в данной предопределенной процедуре установить статус возврата — 0, то ручное редактирование не будет отменено (будет оставлено).

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриОтмененИсправления(Запись)
   // Расчетчик — элемент справочника со списком расчетчиков
   Если Расчетчик.ПравоНаОтмену = Да Тогда
      // отреагируем на это действие
      ПересчитатьСотрудника(Запись.Объект);
   Иначе
      // иначе — нельзя!
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, ПриИсправленииРезультата

ПриРасчете

Предопределенная процедура, вызываемая из формы журнала расчетов при расчете записи журнала расчетов, всего объекта расчета или всех записей по документу.

Синтаксис:

ПриРасчете(<ОбъектРасчета>)

Англоязычный синоним:

OnCalculating

Параметры:

<ОбъектРасчета>	Запись журнала расчетов или элемент справочника, являющийся объектом расчета, или документ, записи которого рассчитываются. Что в данный момент передается системой при вызове процедуры — зависит от выполняемого действия: расчет одной записи, расчет объекта или расчет документа.

Описание:

Вызов предопределенной процедуры ПриРасчете на исполнение производится системой 1С:Предприятие неявно при выполнении одной из трех команд «Рассчитать запись», «РассчитатьОбъект» или «Рассчитать документ» меню «Действия» журнала расчетов. Если в данной предопределенной процедуре установить статус возврата — 0, то расчет проводиться не будет.

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриРасчете(Объект)
   // определим, что считаем?
   Если ТипЗначения(Объект) = 12 Тогда
      // имеем дело с документом
      Если Объект.Автор <> Расчетчик Тогда
         // если автор данного документа не текущий расчетчик, то
         // запретим расчет чужого документа!
         СтатусВозврата(0);
      КонецЕсли;
   ИначеЕсли ТипЗначения(Объект) = 11 Тогда
      // а это — элемент справочника
      Если Объект.Владелец <> Расчетчик Тогда
         // если рассчитываемый объект лежит в папке текущего расчетчика
         // считать можно иначе — нет
         СтатусВозврата(0);
      КонецЕсли;
   Иначе
      // иначе — считаем одну запись
```

```
      Если (Объект.Документ.Автор <> Расчетчик) ИЛИ
            (Объект.Объект.Владелец <> Расчетчик) Тогда
         //не дадим считать запись чужого документа или объекта
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыбореВладельца

Предопределенная процедура выбора элемента справочника по которому будут выведены расчеты.

Синтаксис:

ПриВыбореВладельца(<Владелед>)

Англоязычный синоним:

OnSetOwner

Параметры:

<Владелец>	Значение устанавливаемого владельца — т. е. элемент справочника по которому будут выведены расчеты.

Описание:

Вызов предопределенной процедуры ПриВыбореВладельца производится в системе 1С:Предприятие при интерактивном выборе владельца журнала расчетов (при интерактивной смене владельца, т. е. смене позиции в справочнике-владельце, которая приводит к смене отображаемых расчетов). Если в данной предопределенной процедуре установить статус возврата — 0, то выбор владельца не будет произведен.

Внимание! Процедура выполняется только в том случае, когда журнал расчетов выводится «по объекту», т. е. в нем отображаются записи по одному объекту расчета. Вызов процедуры происходит в момент смены текущего элемента в «главном» справочнике, для которого создан журнал расчетов.

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриВыбореВладельца(Владелец)
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Если Владелец = ЗапрещенныйВладелец Тогда
         Предупреждение("Нельзя изменять объект расчетов;", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриУстановкеОтбора

Предопределенная процедура, вызываемая при установке отбора в форме журнала расчетов.

Синтаксис:

ПриУстановкеОтбора(<ИмяОтбора>, <ЗначениеОтбора>)

Англоязычный синоним:

OnSetSelection

Параметры:

<ИмяОтбора>	Строковое значение. Наименование графы отбора, которая выбрана пользователем при интерактивной установке отбора.
<ЗначениеОтбора>	Значение отбора, которое выбрано пользователем при интерактивной установке отбора.

Описание:

Вызов предопределенной процедуры ПриУстановкеОтбора производится системой 1С:Предприятие неявно при интерактивной попытке установить отбор записей в журнале расчетов. 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 только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриУстановкеОтбора(ИмяОтбора, ЗначОтбора)
   Если НазваниеНабораПрав() = "Расчетчик" Тогда
      Если (ИмяОтбора = "Владелец") И (ЗначОтбора <> ТекущийРасчетчик) Тогда
         Предупреждение("Недостаточно прав!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, УстановитьОтбор, ПолучитьОтбор

ПриУстановкеГраницыПросмотра

Предопределенная процедура, вызываемая при установке границы (по периоду регистрации записи) просматриваемых в журнале расчетов записей.

Синтаксис:

ПриУстановкеГраницыПросмотра(<Период>)

Англоязычный синоним:

OnSetViewBound

Параметры:

<Период>	Значение типа «Период журнала расчетов». Период, устанавливаемый пользователем как граница просмотра записей.

Описание:

Вызов этой предопределенной процедуры производится системой 1С:Предприятие неявно при интерактивной попытке установить другую границу просмотра записей в журнале расчетов. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя просматривать расчеты определенной «давности»), установка не будет выполнена.

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриУстановкеГраницыПросмотра(Период)
   Если НазваниеНабораПрав() = "Расчетчик" Тогда
      ТП = ТекущийПериод();
      // не позволим смотреть более чем на три периода назад
      МинПер = ТП.ПрибавитьПериод(-3);
      Если (Период.ДатаНачала<МинПер.ДатаНачала) Тогда
         Предупреждение("Недостаточно прав 1", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, ГраницаПросмотра

ПриУстановкеПредставления

Предопределенная процедура, вызываемая при установке режима представления журнала расчетов (по всем объектам расчетов, по одному объекту расчета, по одному документу расчета).

Синтаксис:

ПриУстановкеПредставления(<Режим>}

Англоязычный синоним:

OnSetPresentation

Параметры:

<Режим>	Числовое значение — режим представления записей журнала расчетов, устанавливаемый пользователем.

Описание:

Вызов этой предопределенной процедуры производится системой 1С:Предприятие неявно при интерактивной попытке установить другой режим представления записей в журнале расчетов. Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю нельзя использовать определенный режим), установка не будет выполнена.

Параметр <Режим> может принимать следующие значения:

1. Записи по всем объектам расчета.
2. Записи по одному объекту расчета.
3. Записи по одному документу-основанию.

Данная предопределенная процедура может располагаться только в модуле формы журнала расчетов.

Пример:

```
Процедура ПриУстановкеПредставления(Режим)
   Если НазваниеНабораПрав() = "Расчетчик" Тогда
      Если (Режим = 1) Тогда
         Предупреждение("Недостаточно прав!", 2);
         СтатусВозврата(0);
      КонецЕсли;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата, УстановитьПредставление

Глава 23
Работа с Видами и Группами расчетов

Контекст работы с видами расчетов и группами расчетов

Так же как константы и регистры, виды расчетов и группы видов расчета являются частью глобального контекста выполнения программы 1С:Предприятие. Таким образом, для использования атрибутов и методов видов расчетов и групп расчетов достаточно писать непосредственно обращение к атрибутам и методам того или иного расчета или группы с использованием ключевого слова «ВидРасчета» или «ГруппаРасчетов». Это исключает необходимость использования функции `СоздатьОбъект` для получения ссылки на соответствующий агрегатный объект.

Англоязычный синоним ключевого слова `ВидРасчета` — `CalculationKind`.

Англоязычный синоним ключевого слова `ГруппаРасчетов` — `CalculationGroup`.

Пример:

```
*
// Узнаем, входит ли вид расчета ОплатаПоОкладу
//в группу ОблагаютсяНалогом
Проверка =
    ВидРасчета.ОплатаПоОкладу.ВходитВГруппу(ГруппаРасчетов.ОблагаютсяНалогом);
*
// Выведем наименование вида расчета
Сообщить(ВидРасчета.Доплата.Наименование);
```

Кроме того, объекты типа «вид расчета» могут сохраняться в базе данных, т. е. реквизиты справочников, документов, журналов расчета могут иметь тип «ВидРасчета» и хранить ссылки на объекты этого типа.

В глобальном контексте программы 1С:Предприятие есть атрибут «ВидРасчета». Этот объект имеет метод `ПолучитьАтрибут`, который позволяет получить доступ к объекту вида расчета конкретного вида по его имени. Т. е. допустимы следующие синтаксические конструкции:

```
Проверка =
   ВидРасчета.ОплатаПоОкладу.ВходитВГруппу(ГруппаРасчетов.ОблагаетсяНалогом);
Проверка =
   ВидРасчета.ПолучитьАтрибут("ОплатаПоОкладу").
                  ВходитВГруппу(ГруппаРасчетов.ОблагаютсяНалогом);
```

Атрибуты видов расчета и групп видов расчета

Код

Код вида расчета или группы видов расчета.

Синтаксис:

`Код`

А**Англоязычный синоним:**

`Code`

Описание:

Атрибут типа «строка».

Представляет собой строку, соответствующую идентификатору, заданному при конфигурировании.

Пример:

```
ВР01 = ВидРасчета.ПоОкладу;
ВР02 = ВидРасчета.АмортизацияС_НДС;
Группа = ГруппаРасчетов.ОблагаютсяНалогом;
Сообщить(ВР01.Код);
Сообщить(ВР02.Код);
Сообщить(Группа.Код);
```

См. также: `Наименование`

Наименование

Наименование группы видов расчета или вида расчета.

Синтаксис:

`Наименование`

Англоязычный синоним:

`Descriptor`

Описание:

Атрибут типа «строка». Представляет собой строку, соответствующую комментарию, заданному при конфигурировании.

Пример:

```
ВР01 = ВидРасчета.ПоОкладу;
ВР02 = Вид Расчета.АмортизацияС_НДС;
Группа = ГруппаРасчетов.ОблагаютсяНалогом;
Сообщить("Это расчет " + ВР01.Наименование);
```

```
Сообщить(ВР02.Наименование + " " + ВР02.Код);
Сообщить(Строка(Группа.Наименование));
```
См. также: Наименование

Атрибуты видов расчета

Очередность

Очередность вида расчета.

Синтаксис:

Очередность

Англоязычный синоним:

Priority

Описание:

Атрибут типа «число». Представляет собой число, соответствующее очередности вида расчета, заданной при конфигурировании.

Понятие очередности используется для упорядочивания записей журнала расчетов. Это помогает упорядочить расчет записей, тем самым выполняя, например, расчет всех начислений раньше, чем расчет базирующихся на них удержаний.

Пример:

```
ВР01 = ВидРасчета.ПоОкладу;
Сообщить("Это расчет "+ВР01.Наименование+"с очередностью "+ВР01.Очередность);
```
См. также: ПриоритетВытеснения

ПриоритетВытеснения

Приоритет вытеснения вида расчета.

Синтаксис:

ПриоритетВытеснения

Англоязычный синоним:

ReplacePriority

Описание:

Атрибут типа «число». Представляет собой число, соответствующее приоритету вытеснения вида расчета, заданному при конфигурировании. Система использует приоритет вытеснения при разрешении ситуации, когда полностью или частично пересекаются периоды действия видов расчета. Например, когда метод журнала расчетов ВвестиРасчет вводит расчет в «занятый» временной интервал.

Для невытесняющих видов расчета значение этого атрибута равно нулю.

Замечание. Атрибут сохранен для поддержания совместимости с предыдущими версиями программы. Вместо использования данного атрибута рекомендуется использовать методы ВытесняетВидРасчета и ВытесняетсяВидомРасчета.

Пример:

```
*
ВР01 = ВидРасчета.ПоОкладу;
Сообщить("Это расчет "+ВР01.Наименование+" с очередностью "+ВР01.Очередность);
*
Процедура ПровестиДокумент()
   ЖрнЗарплата.ВыбратьЗаписиПоОбъекту(Сотрудник, Начало, Окончание};
   Можно = 1;
   Пока ЖрнЗарплата.ПолучитьЗапись() > 0 Цикл
      Если ЖрнЗарплата.ВидРасч.ПриоритетВытеснения >=
           ВидРасчета.ОплатаБЛ.ПриоритетВытеснения Тогда
         Можно = 0;
      КонецЕсли;
   КонецЦикла;
   Если Начало > Окончание Тогда
      Можно = 0;
   КонецЕсли;
   Если Можно = 1 Тогда
      ЖрнЗарплата.ВвестиРасчет(Сотрудник, ВидРасчета.ОплатаБЛ, Начало,
                    Окончание, 0);
      Если Начало < ЖрнЗарплата.НачалоТекущегоПериода() Тогда
         Перерасчет(ГруппаРасчетов.ПересчДляБЛ, Сотрудник, ТекущийДокумент(),
                     Начало, Окончание);
      КонецЕсли;
   Иначе
      Предупреждение("Некорректный больничный лист!
                  | Проверьте и исправьте даты начала и окончания.
```

```
                    | После этого закройте документ, повторно откройте
                    | и проведите его");
        НеПроводитьДокумент();
    КонецЕсли;
КонецПроцедуры
```

См. также: методы журнала расчетов ВвестиРасчет, ЗаписатьРасчет

Методы видов расчета

ПолучитьАтрибут

Получить доступ к объекту вида расчета конкретного вида по его имени.

Синтаксис:

ПолучитьАтрибут(<ИмяВидаРасчета>)

Англоязычный синоним:

GetAttrib

Параметры:

<ИмяВидаРасчета>	Строковое выражение, содержащее идентификатор вида расчета, как он задан в конфигураторе.

Возвращаемое значение:

Объект вид расчета.

Описание:

Метод ПолучитьАтрибут позволяет получить доступ к объекту вида расчета конкретного вида по его имени, как оно задано в конфигураторе.

Этот метод применяется только к объекту глобально контекста ВидРасчета.

Пример:

```
Начисл = ВидРасчета.ПолучитьАтрибут("Начисления");
```

ВходитВГруппу

Определяет входимость вида расчетов в группу видов расчета.

Синтаксис:

ВходитВГруппу(<Группа>)

Англоязычный синоним:

BelongsToGroup

Параметры:

<Группа>	Группа расчетов, входимость в которую проверяется.

Возвращаемое значение:

Число: 1 — если вид расчета входит в группу <Группа>; 0 в противном случае.

Описание:

Этот метод предназначен для определения входимости вида расчета в конкретную группу расчетов.

Пример:

```
*
// В журнале расчетов зарплаты посчитаем все начисления
// для текущего сотрудника
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
// Начало и конец текущего периода журнала
НТП = ЖР.НачалоТекущегоПериода();
КТП = ЖР.КонецТекущегоПериода();
Группа = Группа.ВсеНачисления;
Всего = 0;
ЖР.ВыбратьЗаписиПоОбъекту(Объект, НТП, КТП);
Пока ЖР.ПолучитьЗапись() = 1 Цикл
    Если (ЖР.ВидРасч.ВходитВГруппу(Группа)=1) Тогда
        Всего = Всего + ЖР.Результат;
    КонецЕсли;
КонецЦикла;
// ...
*
Пока ЖрнЗарплата.ПолучитьЗапись() > 0 Цикл
    Если ЖрнЗарплата.ВидРасч = ВидРасчета.РайонныйКоэФйициент Тогда
        СуммаРайонн = СуммаРайонн + ЖрнЗарплата.Результат;
    КонецЕсли;
    Если ЖрнЗарплата.ВидРасч = ВидРасчета.СевернаяНадбавка Тогда
        СуммаСеверн = СуммаСеверн + ЖрнЗарплата.Результат;
    КонецЕсли;
    Если ЖрнЗарплата.ВидРасч.ВходитВГруппу(Группа) = 1 Тогда
        Сумма = Сумма + ЖрнЗарплата.Результат;
        Если ТипРасчета = 1 Тогда
```

```
            Дней = Дней + КалендОтп.Дней(ЖрнЗарплата.ДатаНачала,
                                          ЖрнЗарплата.ДатаОкончания);
         Иначе
            Дней = Дней + ЖрнЗарплата.Дни;
         КонецЕсли;
      КонецЕсли;
КонецЦикла;
```

См. также: СодержитВидРасчета

Выбран

Возвращает признак того, выбран конкретный вид расчета или нет.

Синтаксис:

Выбран()

Англоязычный синоним:

Selected

Возвращаемое значение:

Число: 1 — если конкретный вид расчета выбран; 0 в противном случае.

Описание:

Этот метод предназначен для определения, не является ли «пустым» значение переменной или реквизита типа «ВидРасчета».

Вид расчета (в отличие от группы расчетов, например) в рамках описываемого встроенного языка — сохраняемый агрегатный тип данных, т. е. объекты этого типа могут сохраняться в информационной базе. Например, реквизиты справочника, документа или журнала расчетов, могут иметь тип «ВидРасчета» и, следовательно, хранить ссылки на конкретные виды расчетов. При этом часто необходимо иметь возможность определить выбрано ли конкретное значение для реквизита (например, справочника или документа) этого типа.

Пример:

```
// Допл — объект типа "группа расчетов"
Допл = ГруппаРасчетов.Доплаты;
ЖЗ = СоздатьОбъект("ЖурналРасчетов.Зарплата");
// Док — ссылка на документ, имеющий реквизит Сотрудник
ЖЗ.ВыбратьЗаписиПоОбъекту(Док.Сотрудник);
Пока ЖЗ.ПолучитьЗапись() = 1 Цикл
   Если ЖЗ.ВидРасч.ВходитВГруппу(Допл) = 1 Тогда
      // предполагается, что документы-основания этих
      // видов расчета имеют реквизит ВР
      Если ЖЗ.Документ.ВР.Выбран() = 0 Тогда
         Сообщить("Не указан конкретный вид расчета в документе!");
      КонецЕсли;
   КонецЕсли;
КонецЦикла;
```

ВытесняетВидРасчета

Определяет, вытесняет ли данный вид расчета, заданный в качестве параметра.

Синтаксис:

ВытесняетВидРасчета(<ВидРасчета>)

Англоязычный синоним:

DisplaceCalc

Параметры:

<ВидРасчета> Вид расчета, для которого проверяется, вытесняется ли он текущим видом расчета.

Возвращаемое значение:

Число: 1 — если вид расчета, метод которого вызывается, вытесняет заданный в качестве параметра <ВидРасчета>; 0 в противном случае.

Описание:

Метод предназначен для определения взаимного влияния видов расчета при вводе их в журнал расчетов.

Пример:

```
// ВР — реквизит документа типа "вид расчета",
// тогда при проведении документа
// Сотр — реквизит документа типа "элемент справочника Сотрудники".
// ДтНач, ДтОконч — реквизиты документа типа "дата"
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ЖР.ВыбратьПоОбъекту(Сотр, ДтНач, ДтОконч);
Пока ЖР.ПолучитьЗапись() = 1 Цикл
   Если ЖР.ВидРасч.ВытесняетВидРасчета(ВР) = 0 Тогда
      // если нашли вид расчета, который мы не сможем вытеснить
      Сообщить("Невозможно корректное проведение документа");
      СтатусВозврата(0);
      Возврат;
   КонецЕсли;
```

```
КонецЦикла;
```
См. также: ВытесняетсяВидомРасчета

ВытесняетсяВидомРасчета

Определяет, вытесняется ли текущий вид расчета заданным видом расчета.

Синтаксис:

```
ВытесняетсяВидомРасчета(<ВидРасчета>)
```

Англоязычный синоним:

```
DisplaceByCalc
```

Параметры:

<ВидРасчета>	Вид расчета, для которого проверяется, вытесняет ли он текущий вид расчета.

Возвращаемое значение:

Число: 1 — если вид расчета, метод которого вызывается, вытесняется видом расчета, заданным в качестве параметра <ВидРасчета>; 0 в противном случае.

Описание:

Метод предназначен для определения взаимного влияния видов расчета при вводе их в журнал расчетов.

См. также: ВытесняетВидРасчета

Методы групп видов расчета

СодержитВидРасчета

Определяет, содержит ли группа вид расчета.

Синтаксис:

```
СодержитВидРасчета(<ВидРасчета>)
```

Англоязычный синоним:

```
ContainCalculationKind
```

Параметры:

<ВидРасчета>	Вид расчета, входимость которого проверяется.

Возвращаемое значение:

Число: 1 — если <ВидРасчета> входит в группу; 0 в противном случае.

Описание:

Метод предназначен для определения того, содержит ли группа заданный вид расчета.

Пример:

```
Входит = ГруппаРасчетов.ОблНалогом.СодержитВидРасчета(ЖР.ВидРасч);
```

См. также: ВходитВГруппу

Количество

Количество видов расчета, включенных в группу видов расчета.

Синтаксис:

```
Количество()
```

Англоязычный синоним:

```
Count
```

Возвращаемое значение:

Целое положительное число — количество видов расчета, входящих в группу.

Описание:

Метод группы видов расчета, который позволяет определить количество видов расчета, включенных в данную группу. Обычно применяется в цикле с методом ПолучитьВидРасчета.

Пример:

```
// модуль выполняется в контексте журнала расчетов
// текущий сотрудник Сотр = Объект;
// Для всех видов удержаний проведем расчет в
// журнале расчетов зарплаты
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
ВсеУд = Группа.ВсеУдержания;
Счетчик = 1;
Пока Счетчик <= ВсеУд.Количество() Цикл
   ЖР.ВвестиРасчет(Сотр, ВсеУд.ПолучитьВидРасчета(Счетчик));
   Счетчик = Счетчик + 1;
КонецЦикла;
```

См. также: ПолучитьВидРасчета

ПолучитьРасчет

Получает ссылку на вид расчета.

Синтаксис:

```
ПолучитьРасчет(<Номер>)
```

Англоязычный синоним:

GetCalculation

Параметры:

<Номер> Номер вида расчета в списке видов расчета данной группы.

Возвращаемое значение:

Агрегатный объект типа «Вид Расчета».

Описание:

Метод позволяет получить ссылку на вид расчета, входящий в данную группу под определенным номером. Обычно применяется в цикле с методом Количество.

Пример:

См. предыдущий пример.

См. также: Количество

Глава 24
Работа с Правилами перерасчета

«Правила перерасчета» являются вспомогательными объектами метаданных компоненты «Расчет». Они предназначены для автоматического отслеживания актуальности результатов проведенных расчетов при вводе новых записей журнала расчетов.

При создании правила перерасчета в конфигураторе определяются виды расчетов, при редактировании которых правило перерасчета «срабатывает», и виды расчетов, которые должны быть перерасчитаны при срабатывании данного правила.

Список видов расчета, на основании которых срабатывает правило перерасчета условно называется «ведущие виды расчета». Список видов расчета, которые должны быть перерасчитаны при срабатывании данного правила условно называется «зависимые виды расчета».

Для организации правильного перерасчета доплат следует в качестве ведущих видов расчета указать те, на основании которых считаются доплаты (например**:** оклад, тариф, сдельная оплата), а в качестве зависимых видов расчета следует указать собственно перерасчитываемые доплаты.

После ввода в систему такого правила перерасчетов журнал расчетов будет вести себя описанным ниже образом. При этом сначала рассмотрим случай для взаимосвязи видов расчета в одном периоде.

Итак, если в журнале расчетов появится новая (в результате проведения документа), исчезнет (при отмене проведения) или будет исправлена существующая запись с одним из «ведущих» видов расчета (в нашем примере оклад, тариф, сдельная оплата), то будет снят признак «Рассчитана» со всех записей, соответствующих доплатам, если найдутся таковые с тем же периодом действия, что и введенная, удаленная или исправленная запись.

Если при этом вводится запись с периодом действия не в текущем расчетном периоде, а в одном из прошлых (например, расчет оклада задним числом за прошлый месяц), то система введет записи-перерасчеты для всех доплат соответствующего прошлого периода.

Правило перерасчета может быть двух типов: перерасчет «по текущему периоду» или «по будущим периодам».

В первом случае перерасчитываются заданные виды расчетов с тем же периодом действия, что и новая введенная запись. Во втором — перерасчитываются записи одного или нескольких будущих расчетных периодов.

Контекст работы с правилами перерасчета

Средства языка предоставляют возможность непосредственного доступа к атрибутам и методам правил перерасчета, объявленных в конфигураторе, в любом программном модуле (все объявленные в конфигураторе правила перерасчета *принадлежат глобальному контексту* конфигурации). Во всех текстах программных модулей для доступа к атрибутам или вызова методов правил перерасчета можно записать соответствующий метод или атрибут через точку после полного имени правила перерасчета.

Полное имя правила перерасчета записывается следующим образом:

`ПравилоПерерасчета.<XXXXX>`,

где <XXXXX> — идентификатор Правила Перерасчета, объявленный в конфигураторе.

Англоязычный синоним ключевого слова `ПравилоПерерасчета` — `RecalculationRule`.

Пример:

```
КолПериодов = ПравилоПерерасчета.Главное.КоличествоПериодов;
ВР = ВидРасчета.ПоОкладу;
ЗависитОтОклада = ПравилоПерерасчета.ПересчДоплат.ИмеетВедущий(ВР);
```

Атрибуты правил перерасчета

Тип

Тип правила перерасчета.

Синтаксис:

`Тип`

Англоязычный синоним:

`Type`

Описание:

Атрибут `Тип` позволяет прочитать/установить значение типа правила перерасчета. Данный атрибут может принимать следующие значения:

- 0 — зависимые виды расчетов должны быть перерасчитаны в том же периоде, что и вводимая запись журнала расчетов;
- 1 — зависимые виды расчетов должны быть перерасчитаны в нескольких периодах, следующих за периодом действия вводимой записи журнала расчетов (количество периодов задано атрибутом `КоличествоПериодов`).

Пример:

```
Тип = ПравилоПерерасчета.Главное.Тип;
```

КоличествоПериодов

Количество перерасчитываемых периодов для правила перерасчета.

Синтаксис:

КоличествоПериодов
Англоязычный синоним:
NumberOfPeriods
Описание:
Атрибут КоличествоПериодов позволяет прочитать/установить количество периодов перерасчета.
Пример:
```
КолПер = ПравилоПерерасчета.Главное.КоличествоПериодов;
```

Методы правил перерасчета

КоличествоВедущих

Возвращает количество «ведущих» видов расчета.
Синтаксис:
КоличествоВедущих()
Англоязычный синоним:
NumberOfLeadings
Возвращаемое значение:
Количество «ведущих» видов расчета.
Описание:
Метод КоличествоВедущих возвращает количество «ведущих» видов расчета (при вводе которых в журнал расчетов необходимо произвести перерасчеты согласно данного правила).
Пример:
```
// выведем все виды расчета от которых зависят доплаты
Для Сч = 1 По ПравилоПерерасчета.Доплаты.КоличествоВедущих() Цикл
   Сообщить(ПравилоПерерасчета.Доплаты.ПолучитьВедущий(Сч));
КонецЦикла;
```

ИмеетВедущий

Установить принадлежность вида расчета к ведущим для правила перерасчета.
Синтаксис:
ИмеетВедущий(<ВидРасчета>)
Англоязычный синоним:
HasLeading
Параметры:
<ВидРасчета> Вид расчета.
Возвращаемое значение:
Число: 1, если данное правило перерасчета имеет заданный вид расчета в качестве ведущего (т. е. при вводе этого вида расчета в журнал расчетов сработает данное правило перерасчета); 0, если данное правило перерасчета не имеет заданный вид расчета в качестве ведущего.
Описание:
Метод ИмеетВедущий позволяет установить принадлежность вида расчета к ведущим для правила перерасчета.
Пример:
```
// зависит ли доплата от премии?
Зависит = ПравилоПерерасчета.Доплаты.ИмеетВедущий(ВидРасчета.Премия);
```

ПолучитьВедущий

Возвращает вид расчета из списка «ведущих» расчетов.
Синтаксис:
ПолучитьВедущий(<НомерВР>)
Англоязычный синоним:
GetLeading
Параметры:
<НомерВР> Числовое выражение — номер вида расчета в списке «ведущих» расчетов.
Возвращаемое значение:
Вид расчета с номером <НомерВР> из списка «ведущих» расчетов.
Описание:
Метод ПолучитьВедущий возвращает вид расчета с номером <НомерВР> из списка «ведущих» расчетов.
Пример:
```
ВидВедРасч = ПравилоПерерасчета.Доплаты.ПолучитьВедущий(Инд);
```

ДобавитьКакВедущий

Добавляет вид расчета в список «ведущих».
Синтаксис:
ДобавитьКакВедущий(<ВидРасчета>)
Англоязычный синоним:
AddAsLeading

Параметры:
<ВидРасчета> Вид расчета.
Описание:
Метод ДобавитьКакВедущий добавляет <ВидРасчета> в список «ведущих».
Пример:
```
ПравилоПерерасчета.Доплаты.ДобавитьКакВедущий(ВидРасчета.Премия);
```

УдалитьВсеВедущие

Удаляет все виды расчета из списка «ведущих».
Синтаксис:
```
УдалитьВсеВедущие()
```
Англоязычный синоним:
```
DeleteAlILeadings
```
Описание:
Метод УдалитьВсеВедущие удаляет все виды расчета из списка «ведущих».
Пример:
```
ПравилоПерерасчета.Доплаты.УдалитьВсеВедущие();
```

КоличествоПодчиненных

Возвращает количество «подчиненных» видов расчета.
Синтаксис:
```
КоличествоПодчиненных()
```
Англоязычный синоним:
```
NumberOfDependents
```
Возвращаемое значение:
Количество «подчиненных» видов расчета.
Описание:
Метод КоличествоПодчиненных возвращает количество «подчиненных» видов расчета (для которых необходимо произвести перерасчеты согласно данного правила).
Пример:
```
Колич = ПравилоПерерасчета.Доплаты.КоличествоПодчиненных();
```

ИмеетПодчиненный

Установить принадлежность вида расчета к «подчиненным» для правила перерасчета.
Синтаксис:
```
ИмеетПодчиненный(<ВидРасчета>)
```
Англоязычный синоним:
```
HasDependent
```
Параметры:
<ВидРасчета> Вид расчета.
Возвращаемое значение:
Число: 1, если данное правило перерасчета имеет заданный вид расчета в качестве подчиненного (т. е. для этого вида расчета в журнале расчетов сработает данное правило перерасчета); 0, если данное правило перерасчета не имеет заданный вид расчета в качестве подчиненного.
Описание:
Метод ИмеетПодчиненный позволяет установить принадлежность вида расчета к подчиненным для правила перерасчета.
Пример:
```
Подч = ПравилоПерерасчета.Доплаты.ИмеетПодчиненный(ВидРасчета.Премия);
```

ПолучитьПодчиненный

Возвращает вид расчета из списка «подчиненных» расчетов.
Синтаксис:
```
ПолучитьПодчиненный(<НомерВР>)
```
Англоязычный синоним:
```
GetDependent
```
Параметры:
<НомерВР> Числовое выражение — номер вида расчета в списке «подчиненных» расчетов.
Возвращаемое значение:
Вид расчета с номером <НомерВР> из списка «подчиненных» расчетов.
Описание:
Метод ПолучитьПодчиненный возвращает вид расчета с номером <НомерВР> из списка «подчиненных» расчетов.
Пример:
```
ВидПодчРасч = ПравилоПерерасчета.Доплаты.ПолучитьПодчиненный(Инд);
```

ДобавитьКакПодчиненный

Добавляет вид расчета в список «подчиненных».

Синтаксис:

`ДобавитьКакПодчиненный(<ВидРасчета>)`

Англоязычный синоним:

`AddAsDependent`

Параметры:

`<ВидРасчета>` Вид расчета.

Описание:

Метод `ДобавитьКакПодчиненный` добавляет `<ВидРасчета>` в список «подчиненных».

Пример:

```
ПравилоПерерасчета.Доплаты.ДобавитьКакПодчиненный(ВидРасчета.Премия);
```

УдалитьВсеПодчиненные

Удаляет все виды расчета из списка «подчиненных».

Синтаксис:

`УдалитьВсеПодчиненные()`

Англоязычный синоним:

`DeleteAllDependents`

Описание:

Метод `УдалитьВсеПодчиненные` удаляет все виды расчета из списка «подчиненных».

Пример:

```
ПравилоПерерасчета.Доплаты.УдалитьВсеПодчиненные();
```

Применять

Устанавливает необходимость применять или не применять все правила перерасчетов при вводе новых записей в журнал расчетов.

Синтаксис:

`Применять(<ФлагПрименения>)`

Англоязычный синоним:

`Use`

Параметры:

`<ФлагПрименения>` Число: 1 — применять; 0 — не применять.

Возвращаемое значение:

Текущее значение флага применения (до исполнения метода).

Описание:

Метод `Применять` устанавливает необходимость применять или не применять все заданные в конфигурации правила перерасчетов при изменении состава или исправлении записей журнала расчетов.

Пример:

```
ПравилоПерерасчета.Применять(1);
```

Глава 25
Работа с Календарями и Праздниками

Объект Календарь в системе 1С:Предприятие предназначен для вычисления временных интервалов на основании произвольных временных графиков. Этот объект может использоваться, например, для учета графика работы предприятия, графика работы внешних организаций, графика учета рабочего времени тех или иных работников предприятия и пр.

Календарь представляет собой объект, который устанавливает для каждой календарной даты некоторое числовое значение, которое может интерпретироваться при работе программы тем или иным образом. Например, в календаре, представляющем собой график работы организации или предприятия, каждому рабочему дню может соответствовать единица, а каждому выходному — ноль. В календаре учета рабочего времени сотрудников предприятия, каждой дате календаря может соответствовать число — продолжительность рабочего дня в часах и т. д.

В глобальном контексте программы 1С:Предприятие есть атрибут "Календари". Этот объект в качестве своих атрибутов имеет значения объектов конкретных видов календарей, заданных в конфигурации. Кроме того, этот объект имеет метод ПолучитьАтрибут, который позволяет получить доступ к объекту календаря конкретного вида по его имени. Т.е. допустимы следующие синтаксические конструкции:

```
Календари.Служащие.Дней(ТекДата, ТекДата + 5);
Календари.ПолучитьАтрибут("Служащие").Дней(ТекДата, ТекДата + 5);
```

Праздники в системе 1С:Предприятие представляет собой объект похожий на календарь, но он заполняется выборочно (не все даты подряд) и в нем могут вводиться и удаляться строки для определенных дат. Данный объект используются как набор исключений при заполнении календарей (см. метод календаря УчитыватьПраздники).

Контекст работы с календарями

Доступ к атрибутам и методам календарей осуществляется, например, через объект, созданный системной функцией СоздатьОбъект с ключевым словом

"Календарь".

Англоязычный синоним ключевого слова "Календарь" — "Calendar".

Пример:

```
// Рассчитать продолжительность месяца в часах по календарю "Служащие"
Календ = СоздатьОбъект("Календарь.Служащие");
Час = Календ.Часов('01.01.96', '31.01.96');
// Рассчитать продолжительность текущего расчетного
// периода журнала расчетов Зарплата по календарю Служащие
Календ = СоздатьОбъект("Календарь.Служащие");
ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
Час = Календ.Часов(ЖР.НачалоТекущегоПериода(), ЖР.КонецТекущегоПериода());
```

Кроме того, объекты типа «календарь» являются сохраняемыми, т. е. реквизиты справочников, документов, журналов расчета могут иметь тип «календарь» и хранить ссылки на объекты этого типа.

Пример:

```
Процедура ПровестиРасчет()
   // ВнСовместитель — реквизит журнала расчетов
   // Категории — периодический реквизит справочника ВнутреннииеСовместители
   // Календарь — реквизит справочника Категории
   Календ = ВнСовместитель.Категория.Получить(ДатаОкончания).Календарь;
   // размер оклада
   Оклад = ВнСовместитель.Оклад.Получить(ДатаОкончания) *
                        ВнСовместитель.Ставка.Получить(ДатаОкончания);
   Дней = Календ.Дней(ДатаНачала, ДатаОкончания);
   ВсегоДней = Календ.Дней(НачалоПериодаПоДате(ДатаНачала),
                           КонецПериодаПоДате(ДатаНачала));
   Если ВсегоДней > 0 Тогда
      Если Документ.Вид() = "НачалоМесяца" Тогда
         Результат = Окр(Оклад * Дней / ВсегоДней);
      Иначе
         Результат = -Окр(Оклад*Дней/ВсегоДней);
      КонецЕсли;
   Иначе
      Результат = 0;
      Сообщить("Неправильно указан календарь (" + Объект.Наименование + ")");
      Сообщить("Возможно, не задана категория работника");
   КонецЕсли;
КонецПроцедуры
```

Контекст работы с праздниками

Доступ к атрибутам и методам праздников осуществляется через объект, создаваемый системной функцией СоздатьОбъект с ключевым словом "Праздники".

Англоязычный синоним ключевого слова "Праздники" — "Holidays".

Атрибуты и методы объекта Календари

<ИмяКалендаря>

Установить значение календаря конкретного вида.

Синтаксис:

<ИмяКалендаря>

Описание:

Атрибут <ИмяКалендаря> задает значение календаря конкретного вида. В тексте программного модуля используется название конкретного вида календаря, как он назван в конфигураторе.

Пример:

```
Календари.Служащие.Дней(ТекДата, ТекДата + 5);
```

ПолучитьАтрибут

Получить значение объекта конкретного вида календаря по идентификатору.

Синтаксис:

ПолучитьАтрибут(<ИмяАтрибута>)

Англоязычный синоним:

GetAttrib

Параметры:

<ИмяАтрибута>	Строковое выражение, содержащее имя конкретного вида календаря, как оно задано в конфигураторе.

Возвращаемое значение:

Значение атрибута <ИмяАтрибута >.

Описание:

Метод ПолучитьАтрибут позволяет получить значение объекта конкретного вида календаря по идентификатору, как оно задано в метаданных.

Пример:

```
Календари.ПолучитьАтрибут("Служащие").Дней(ТекДата, ТекДата - 5);
```

Атрибуты календарей и праздников

Дата

Дата текущей строки календаря/праздника.

Синтаксис:

Дата

Англоязычный синоним:

Date

Описание:

Атрибут Дата (только для чтения) содержит дату текущей (на которой позиционирована выборка) строки календаря/праздника.

Значение

Значение текущей строки календаря/праздника

Синтаксис:

Значение

Англоязычный синоним:

Value

Описание:

Атрибут Значение (доступно чтание и запись) содержит значение текущей (на которой позиционирована выборка) строки календаря/праздника — число с двумя знаками после запятой.

Общие методы календарей и праздников

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. Доступные имена атрибутов — "Дата" и "Значение".

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Календ.СледующаяДата() = 1 Цикл
   Сообщить("" + Календ.ПолучитьАтрибут("Дата") + "; ");
   // ...
КонецЦикла;
```

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. Доступные имена атрибутов — "Дата" и "Значение".

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Календ.СледующаяДата() = 1 Цикл
   Календ.УстановитьАтрибут("Значение", 1);
   // ...
КонецЦикла;
```

Методы календарей

Выбран

Возвращает признак того, выбран конкретный календарь или нет.

Синтаксис:

```
Выбран()
```

Англоязычный синоним:

```
Selected
```

Возвращаемое значение:

Число: 1, если конкретный календарь выбран; 0 в противном случае.

Описание:

Этот метод предназначен для определения, не является ли «пустым» значение переменной или реквизита типа «Календарь».

Календарь в рамках описываемого встроенного языка — сохраняемый агрегатный тип данных, т. е. объекты этого типа могут сохраняться в информацион-йой базе. Например, реквизиты справочника, документа или журнала расчетов, Могут иметь тип «Календарь» и, следовательно, хранить ссылки на конкретные календари. При этом часто необходимо иметь возможность определить выбрано ли конкретное значение для реквизита (например, справочника или документа) этого типа.

Пример:

```
// реквизит справочника Календ имеет тип "Календарь"
Спр = СоздатьОбъект("Справочник.Категории");
Спр.ВыбратьЭлементы();
Пока Спр.ПолучитьЭлемент() = 1 Цикл
   Если Спр.Календ.Выбран() = 0 Тогда
      Сообщить("Не указан конкретный календарь! ");
   КонецЕсли;
КонецЦикла;
```

ВыбратьДаты

Открывает выборку записей календаря.

Синтаксис:

```
ВыбратьДаты(<ДатаНачала>, <ДатаОкончания>)
```

Англоязычный синоним:

```
SelectDates
```

Параметры:

<ДатаНачала>	Дата начала периода выборки.
<ДатаОкончания>	Дата окончания периода выборки.

Возвращаемое значение:

Число: 1 — выборка открыта и в ней есть хоть одна запись; 0 — не обнаружено записей.

Описание:

Метод ВыбратьДаты открывает выборку записей календаря. Выбираются все элементы с даты начала по дату окончания включительно.

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Календ.СледующаяДата() = 1 Цикл
   // ...
КонецЦикла;
```

См. также: СледующаяДата

СледующаяДата

Выбирает следующий день календаря в выборке.

Синтаксис:

```
СледующаяДата()
```

Англоязычный синоним:

```
NextDate
```

Возвращаемое значение:

Число: 1 — получена очередна язапись; 0 — не обнаружено очередной записи, т. е. конец выборки.

Описание:

Выбирает следующий день календаря (позиционируется на очередной записи выборки) в порядге дат.

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Календ.СледующаяДата() = 1 Цикл
   //. . .
КонецЦикла;
```

См. также: ВыбратьДаты

Дней

Получить количество «ненулевых» дней за период.

Синтаксис:

```
Дней(<ДатаНачала>, <ДатаОкончания>)
```

Англоязычный синоним:

```
Days
```

Параметры:

<ДатаНачала>	Дата начала периода, в котором определяется количество дней.
<ДатаОкончания>	Дата окончания периода, в котором определяется количество дней.

Возвращаемое значение:

Целое положительное число — количество дней.

Описание:

Метод предназначен для определения количества календарных дней, которым соответствует ненулевое числовое значение. <ДатаНачала> и <ДатаОкончания> включаются в рассматриваемый период. Следовательно, если даты начала и окончания совпадают, метод возвратит ноль или единицу в зависимости от значения числа, соответствующего заданной дате. Если <ДатаНачала> больше, чем <ДатаОкончания>, будет выведено сообщение об ошибке.

Пример:

```
// Расчет оплаты по окладу
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
// отработал...
Дни = Календ.Дней(ДатаНачала, ДатаОкончания);
// должен был бы работать...
Норма = Календ.Дней(НачалоПериодаПоДате(ДатаНачала),
                    КонецПериодаПоДате(ДатаНачала));
// получим оклад сотрудника на дату начала
// действия данного расчета
Оклад = Объект.Оклад.Получить(ДатаНачала);
// результат
Результат = Дни * Оклад / Норма;
```

См. также: Часов

Часов

Получить сумму числовых значений дней за определенный период.

Синтаксис:

`Часов(<ДатаНачала>, <ДатаОкончания>)`

Англоязычный синоним:

`Hours`

Параметры:

`<ДатаНачала>`	Дата начала периода, в котором определяется сумма числовых значений, интерпретируемая как количество часов.
`<ДатаОкончания>`	Дата окончания периода, в котором определяется сумма числовых значений, интерпретируемая как количество часов.

Возвращаемое значение:

Положительное число — сумма числовых значений соответствующих тем дням календаря, которые попадают в заданный период.

Описание:

Метод предназначен для суммирования числовых значений по календарю за промежуток времени с даты начала по дату окончания. Метод удобен при интерпретации числовых значений календаря как продолжительности рабочих дней в часах (что отражается в названии метода). `<ДатаНачала>` и `<ДатаОкончания>` включаются в рассматриваемый период. Если `<ДатаНачала>` больше, чем `<ДатаОкончания>`, будет выведено сообщение об ошибке.

Пример:

```
// Расчет оплаты по часовому тарифу
// Календарь рабочих
Календ = СоздатьОбъект("Календарь.Рабочие");
// отработал...
Чс = Календ.Часов(ДатаНачала, ДатаОкончания);
// должен был бы работать...
Норма = Календ.Часов(НачалоПериодаПоДате(ДатаНачала),
                     КонецПериодаПоДате(ДатаНачала));
// получим тариф сотрудника на дату начала действия данного расчета
Тариф = Объект.Тариф.Получить(ДатаНачала);
// результат
Результат = Чс * Тариф / Норма;
```

См. также: `Дней`

Автозаполнение

Выполняет автозаполнение календаря в заданном периоде.

Синтаксис:

`Автозаполнение(<ДатаНачала>, <ДатаОкончания>)`

Англоязычный синоним:

`Autolayout`

Параметры:

`<ДатаНачала>`	Дата начала периода автозаполнения.
`<ДатаОкончания>`	Дата окончания периода автозаполнения.

Возвращаемое значение:

Число: 1 — получилось; 0 — не получилось.

Описание:

Метод `Автозаполнение` выполняет автозаполнение календаря в заданном периоде. При этом старые данные (в т.ч. «ручные» исправления) в указанном периоде пропадают.

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.УчитыватьПраздники(1);
Результат = Календ.Автозаполнение(ДатаНачала, ДатаОкончания);
```

См. также: `УчитыватьПраздники`

УчитыватьПраздники

Установка флага учета праздников при автозаполнении.

Синтаксис:

`УчитыватьПраздники(<Флаг>)`

Англоязычный синоним:

`UseHolidays`

Параметры:

`<Флаг>`	Необязательный параметр. Число: 1 — учитываются праздники при автозаполнении; 0 — не учитываются праздники при автозаполнении.

Возвращаемое значение:

Возвращает текущую установку. Число: 1 — учитываются праздники при автозаполнении; 0 — не учитываются праздники при автозаполнении.

Описание:

Метод УчитыватьПраздники выполняет установку флага учета праздников при автозаполнении.

Пример:

```
// Календарь служащих
Календ = СоздатьОбъект("Календарь.Служащие");
Календ.УчитыватьПраздники(1);
Результат = Календ.Автозаполнение(ДатаНачала, ДатаОкончания);
```

См. также: Автозаполнение

ПолучитьДату

Получить дату календаря.

Синтаксис:

ПолучитьДату(<ДатаНачала>, <Количество>)

Англоязычный синоним:

GetDate

Параметры:

<ДатаНачала>	Дата начала периода, от которой будут отсчитываться дни.
<Количество>	Количество дней, отсчитываемых от <ДатаНачала>.

Возвращаемое значение:

Значение типа «дата».

Описание:

Метод предназначен для определения даты, отстоящей от заданной даты на известное количество дней. В отличие от простого прибавления числа к дате, результатом которого является дата, смещенная на заданное количество дней, данный метод календаря производит определение итоговой даты, пропуская дни календаря, для которых задано нулевое значение. Это значит, например, что если в конфигурации присутствует календарь, представляющий собой график работы предприятия, то данный метод позволит легко определить дату, отстоящую от заданной на определенное количество «рабочих» дней (т. е. дней, для которых задано ненулевое числовое значение).

Пример:

```
Процедура ПровестиДокумент()
   Если Число(Окончание) = 0 Тогда
      // календарь
      Календ = Сотрудник.Категория.Получить(Начало).Календарь;
      // при расчете отпуска, запишем дату его окончания
      Календ = СоздатьОбъект("Календарь.Отпуск");
      Окончание = Календ.ПолучитьДату(Начало, КоличДней);
   КонецЕсли;
   Если Расчет <> ВидРасчета.ОтпускПоУходуЗаРебенком Тогда
      Если Окончание > ЖрнЗарплата.КонецТекущегоПериода() Тогда
         РасчетВперед(ГруппаРасчетов.ПересчДляОтпуска, Сотрудник,
                      ТекущийДокумент());
      КонецЕсли;
      ЖрнЗарплата.ВвестиРасчет(Сотрудник, Расчет, Начало, Окончание, 0);
   Иначе
      ЖрнЗарплата. ВвестиРасчет (Сотрудник, Расчет,
             Максимум(Начало, ЖрнЗарплата.НачалоТекущегоПериода()),
                  Минимум(Окончание, ЖрнЗарплата.КонецТекущегоПериода()), 0);
      Зарегистрировать(Контекст, ТекущийДокумент());
   КонецЕсли;
КонецПроцедуры
```

Методы праздников

Новый

Ввести новый «праздничный» день.

Синтаксис:

Новый(<Дата>, <Значение>)

Англоязычный синоним:

New

Параметры:

<Дата>	Дата нового праздника.
<Значение>	Положительное число с двумя знаками после запятой (не более 100). Поскольку объект «Праздники», как правило, используется для ввода выходных праздничных дней по календарям, то значение, как правило, задают равным нулю.

Описание:

Метод Новый вводит и записывает новую запись (новый «праздничный» день).

Пример:

```
// Праздничные дни
Празд = СоздатьОбъект("Праздники");
Празд.Новый('01.05.99', 0)
```

Удалить

Удаляет «праздничный» день.

Синтаксис:

```
Удалить(<Дата>)
```

Англоязычный синоним:

```
Delete
```

Параметры:

`<Дата>`	Дата удаляемой строки (праздника).

Описание:

Метод `Удалить` позволяет удалить существующую запись («праздничный» день). Если строка с такой датой не найдена, то метод ничего не делает. Возвращаемого значения нет.

Пример:

```
// Праздничные дни
Празд = СоздатьОбъект("Праздники");
Празд.Удалить('01.05.99');
```

ВыбратьДаты

Открывает выборку записей праздников.

Синтаксис:

```
ВыбратьДаты(<ДатаНачала>, <ДатаОкончания>)
```

Англоязычный синоним:

```
SelectDates
```

Параметры:

`<ДатаНачала>`	Дата начала периода выборки.
`<ДатаОкончания>`	Дата окончания периода выборки.

Возвращаемое значение:

Число: 1 — выборка открыта и в ней есть хоть одна запись; 0 — не обнаружено записей.

Описание:

Метод `ВыбратьДаты` открывает выборку записей праздников. Выбираются все элементы с даты начала по дату окончания включительно.

Пример:

```
// Праздничные дни
Празд = СоздатьОбъект("Праздники");
Празд.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Празд.СледующаяДата() = 1 Цикл
   // ...
КонецЦикла;
```

См. также: `СледующаяДата`

СледующаяДата

Выбирает следующий день праздников в выборке.

Синтаксис:

```
СледующаяДата()
```

Англоязычный синоним:

```
NextDate
```

Возвращаемое значение:

Число: 1 — получена очередна язапись; 0 — не обнаружено очередной записи, т. е. конец выборки.

Описание:

Выбирает следующий день праздников (позиционируется на очередной записи выборки) в порядке дат.

Пример:

```
// Праздничные дни
Празд = СоздатьОбъект("Праздники");
Празд.ВыбратьДаты(ДатаНачала, ДатаОкончания);
Пока Празд.СледующаяДата() = 1 Цикл
   // ...
КонецЦикла;
```

См. также: `ВыбратьДаты`

Глава 26
Работа с последовательностями документов

«Последовательности документов» являются вспомогательными объектами метаданных для компонент «Оперативный учет» и «Бухгалтерский учет». Они предназначены для обеспечения проведения определенных документов в непрерывной хронологической последовательности.

При создании последовательности определяется, какие виды документов входят в данную последовательность, а также виды движений, влияющих на последовательность. Виды движений определяют, какие из изменений итогов будут влиять на необходимость перепроведения документов данной последовательности, то есть итоги каких механизмов учета используются документами данной последовательности при проведении.

Например, для организации правильного списания товаров следует в качестве движения, влияющего на последовательность указать регистр оперативного учета, на котором ведется стоимостной учет товаров или, если учет ведется на бухгалтерских счетах, указать счет, на котором ведется учет товаров. В качестве документов, на проведение которых будет влиять данная последовательность, следует указать те виды документов, которые будут анализировать при проведении указанный регистр или остатки по указанному счету. Например, такими документами могут быть расходные накладные, накладные на передачу на реализацию и т. п.

Подробнее о последовательностях документов следует читать в книге «Руководство по конфигурированию и администрированию».

Контекст работы с последовательностями

Несмотря на то, что механизм отслеживания и восстановления последовательности документов отрабатывает системой автоматически, средства языка предоставляют возможность непосредственного доступа к методам последовательностей. Доступ к методам возможен в любом программном модуле (все объявленные в конфигураторе последовательности *принадлежат глобальному контексту* конфигурации). Во всех текстах программных модулей вызовы методов последовательностей можно записывать просто через точку после полного имени последовательности. Полное имя последовательности записывается следующим образом:

`Последовательность.<XXXXX>`

где <XXXXX> — идентификатор последовательности, объявленный в конфигураторе.

Англоязычный синоним ключевого слова `Последовательность` — `Sequence`.

Пример:

```
Последовательность.УправленческийУчет.Установить(ТекущийДокумент())
```

Методы последовательностей

ПолучитьПозицию

Получить позицию Границы Последовательности.

Синтаксис:

```
ПолучитьПозицию()
```

Англоязычный синоним:

```
GetPosition
```

Возвращаемое значение:

32-х символьное строковое значение позиции Границы Последовательности.

Описание:

Метод `ПолучитьПозицию` возвращает позицию Границы Последовательности.

Пример:

```
ГП = Последовательность.УправленческийУчет.ПолучитьПозицию();
```

Получить

Получить строковое представление Границы Последовательности.

Синтаксис:

```
Получить()
```

Англоязычный синоним:

```
Get
```

Возвращаемое значение:

Строковое представление Границы Последовательности.

Описание:

Метод `Получить` возвращает строковое представление Границы Последовательности.

Пример:

```
ПредставлениеГП = Последовательность.УправленческийУчет.Получить();
```

ПолучитьДокумент

Получить документ Границы Последовательности.

Синтаксис:

```
ПолучитьДокумент()
```

Англоязычный синоним:

GetDocument

Возвращаемое значение:
Документ Границы Последовательности.

Описание:
Метод ПолучитьДокумент возвращает документ Границы Последовательности.

Пример:
```
ПокумГП = Последовательность.УправленческийУчет.ПолучитьДокумент();
```

ПолучитьДату

Получить дату Границы Последовательности.

Синтаксис:
ПолучитьДату()

Англоязычный синоним:
GetDate

Возвращаемое значение:
Дата Границы Последовательности.

Описание:
Метод ПолучитьДату возвращает дату Границы Последовательности.

Пример:
```
ДатаГП = Последовательность.УправленческийУчет.ПолучитьДату();
```

ПолучитьВремя

Получить время Границы Последовательности.

Синтаксис:
ПолучитьВремя(<Часы>, <Минуты>, <Секунды>)

Англоязычный синоним:
GetTime

Параметры:

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

Возвращаемое значение:
Строковое значение времени Границы Последовательности в виде "ЧЧ.ММ.СС".

Описание:
Метод ПолучитьВремя возвращает время Границы Последовательности.

Пример:
```
Функция ВремяГП()
   Перем Ч;
   Перем М;
   Перем С;
   ВремяГП = Последовательность.УправленческийУчет.ПолучитьВремя();
   Возврат "Время ГП в " + Ч + " час " + М + " мин. " + С + " с";
КонецФункции
```

Установить

Установить ГП на новую дату документ или позицию.

Синтаксис:
Установить(<ПоложениеГП>)

Англоязычный синоним:
Set

Параметры:

<ПоложениеГП>	Выражение типа «дата», «документ» или «позиция документа», на который устанавливается ГП.

Описание:
Метод Установить изменяет положение Границы Последовательности на начало даты или документа.

Внимание. Следует особо обратить внимание, что механизм последовательности документов отрабатывает системой автоматически, поэтому метод Установить можно применять только в особых случаях, в основном, чтобы ГП принудительно отодвинуть назад, так как при установке вперед документы не перерасчитываются.

Пример:
```
Последовательность.УправленческийУчет.Установить(ТекущийДокумент());
```

Сравнить

Сравнить ГП с датой, документом или позицией.

Синтаксис:

Сравнить(<Докум>)

Англоязычный синоним:

Compare

Параметры:

<Докум> Выражение типа «дата», «документ» или «позиция документа», с которым сравнивается ГП.

Возвращаемое значение:

Число: -1 (минус единица), если ГП меньше (раньше); 0 если равны; 1 если ГП больше (позже).

Описание:

Метод Сравнить сравнивает ГП с датой, документом или позицией документа.

Пример:

```
Если Последовательность.УправленческийУчет.Сравнить(Док) = 1 Тогда
   Последовательность.УправленческийУчет.Установить(Док);
КонецЕсли;
```

ПринадлежитПоследовательности

Определить, принадлежит ли последовательности заданный документ или вид, заданный строкой.

Синтаксис:

ПринадлежитПоследовательности(<Докум>}

Англоязычный синоним:

BelongSequence

Параметры:

<Докум> Выражение типа «документ» или «строка», определяющая вид документа.

Возвращаемое значение:

Число: 1 — если документ принадлежит последовательности; 0 — если не принадлежит.

Описание:

Метод ПринадлежитПоследовательности позволяет определить, принадлежит ли последовательности заданный документ или вид, заданный строкой.

Пример:

```
Если Последовательность.УпрУчет.ПринадлежитПоследовательности(Док) = 1 Тогда
   Если Последовательность.УпрУчет.Сравнить(Док) = 1 Тогда
      Последовательность.УпрУчет.Установить(Док);
   КонецЕсли;
КонецЕсли;
```

Проверить

Проверяет, является ли последовательность непрерывной от ГП до заданного документа.

Синтаксис:

Проверить(<Докум>)

Англоязычный синоним:

Validate

Параметры:

<Докум> Выражение типа «дата», «документ» или «позиция документа», от которой проверяется непрерывность последовательности.

Возвращаемое значение:

Число: 1 если последовательность непрерывна; 0 если есть проведенные документы между ГП и документом.

Описание:

Метод Проверить проверяет, есть ли между ГП и данной точкой проведенные документы входящие в даную последовательность (т. е. является ли последовательность непрерывной от ГП до этой точки).

Пример:

```
Если Последовательность.УправленческийУчет.Проверить(Док) = 1 Тогда
   Последовательность.УправленческийУчет.Установить(Док);
КонецЕсли;
```

Глава 27
Работа с объектом Периодический

Для работы с периодическими реквизитами справочников и периодическими константами в системе используется специальный агрегатный тип данных — «Периодический». Объекты данного типа предназначены для возможности записи, редактирования и удаления значений периодических реквизитов справочников и периодических констант непосредственно из программного модуля, без необходимости прибегать к интерактивным операциям.

Контекст работы с объектом Периодический

Во всех программных модулях доступ к атрибутам и вызов методов объекта *Периодический* может выполняться только при помощи переменной со ссылкой на объект этого типа. Объект создается при помощи функции `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

При создании объекта данного типа функции `СоздатьОбъект` в качестве параметра передается ключевое слово `"Периодический"`.

Англоязычный синоним ключевого слова `Периодический` — `Periodic`.

Пример:

```
ВремРеквизиты = СоздатьОбъект("Периодический");
```

Атрибуты объекта Периодический

Значение

Значение периодического реквизита справочника или константы.

Синтаксис:

`Значение`

Англоязычный синоним:

`Value`

Описание:

Атрибут `Значение` предоставляет доступ к значению выбранного периодического реквизита справочника или константы.

Пример:

```
Вал = СоздатьОбъект("Справочник.Валюты");
// Позиционируем созданный объект Вал по известному коду
Вал.НайтиПоКоду(1);
Если Вал.Выбран() = 1 Тогда
   Доллар = Вал.ТекущийЭлемент();
Иначе
   Предупреждение("Не найдена валюта!");
   Возврат;
КонецЕсли;
ПерВал = СоздатьОбъект("Периодический");
ПерВал.ИспользоватьОбъект("Текущ_курс".Доллар);
ПерВал.ОбратныйПорядок(1);
ПерВал.ВыбратьЗначения();
Пока ПерВал.ПолучитьЗначение() = 1 Цикл
   Курс = ПерВал.Значение;
   ДатаКурса = ПерВал.ДатаЗнач;
КонецЦикла;
```

ДатаЗнач

Дата значения периодического реквизита справочника или константы.

Синтаксис:

`ДатаЗнач`

Англоязычный синоним:

`DateVal`

Описание:

Атрибут `ДатаЗнач` предоставляет доступ к дате значения выбранного периодического реквизита справочника или константы.

Пример:

См. предыдущий пример

Методы объекта Периодический

ИспользоватьОбъект

Задать объект применения.

Синтаксис:
`ИспользоватьОбъект(<ИмяРеквизита>, <Объект>)`
Англоязычный синоним:
`UseObject`
Параметры:

`<ИмяРеквизита>`	Строковое выражение, задающее название периодического реквизита справочника или название периодической константы, как они названы в конфигураторе.
`<Объект>`	Необязательный параметр. Значение элемента справочника, для которого задается применение объекта «Периодический». Данный параметр требуется задавать только в случае, если `<ИмяРеквизита>` — периодический реквизит справочника.

Возвращаемое значение:
Число: 1 — если вызов метода закончился успешно, 0 — если нет.
Описание:
Метод `ИспользоватьОбъект` задает соответствие созданного ранее объекта типа «Периодический» тому периодическому реквизиту справочника или периодической константе, для которой он будет применяться. Если параметр `<ИмяРеквизита>` не задан (пустая строка), а параметр `<Объект>` задает элемент справочника, то выборка будет осуществляться по всем периодическим реквизитам справочника.
Пример:

```
Вал = СоздатьОбъект("Справочник.Валюты");
// Позиционируем созданный справочник Валюты по известному коду
Вал.НайтиПоКоду(1);
Если Вал.Выбран() = 1 Тогда
   Доллар = Вал.ТекущийЭлемент();
Иначе
   Предупреждение{"Не найдена валюта!");
   Возврат;
КонецЕсли;
ПерВал = СоздатьОбъект("Периодический");
ПерВал.ИспользоватьОбъект("Текущ_курс", Доллар);
ПерВал.ОбратныйПорядок(1);
ПерВал.ВыбратьЗначения();
Пока ПерВал.ПолучитьЗначение() = 1 Цикл
   Курс = ПерВал.Значение;
   ДатаКурса = ПерВал.ДатаЗнач;
КонецЦикла;
```

НазначитьТип

Назначить тип для периодического объекта неопределенного типа.
Синтаксис:
`НазначитьТип(<ИмяТипа>, <Длина>, <Точность>)`
Англоязычный синоним:
`SetType`
Параметры:

`<ИмяТипа>`	Строковое выражение — название типа данных, который назначается периодическому реквизиту справочника или периодической константе неопределенного типа. 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 периодического реквизита справочника или периодической константе неопределенного типа.
Пример:

```
Тов = СоздатьОбъект("Справочник.Номенклатура");
// Позиционируем созданный справочник по известному коду
Тов.НайтиПоКоду(51);
Если Тов.Выбран() = 1 Тогда
   ВыбТМЦ = Тов.ТекущийЭлемент();
Иначе
   Предупреждение("Не найден товар!");
   Возврат;
КонецЕсли;
ПерТМЦ = СоздатьОбъект("Периодический");
ПерТМЦ.ИспользоватьОбъект("ТМЦ", ВыбТМЦ);
ПерТМЦ.НазначитьТип("Справочник.Товары");
```

ЗначениеНаДату

Получить актуальное значение на заданную дату.

Синтаксис:

```
ЗначениеНаДату(<Дата>)
```

Англоязычный синоним:

```
ValueOnDate
```

Параметры:

<Дата>	Выражение, задающее значение даты, на которую требуется получить периодический реквизит справочника или периодическую константу.

Возвращаемое значение:

Полученное актуальное значение на заданную дату.

Описание:

С помощью метода ЗначениеНаДату можно получить значение, актуальное на заданную дату. Причем текущая позиция выборки (см. ВыбратьЗначение) не сдвигается и не сбрасывается.

Пример:

```
К = ПерВал.ЗначениеНаДату(ДатаП);
```

НайтиЗначение

Найти значение на заданную дату.

Синтаксис:

```
НайтиЗначение(<Дата>, <Режим>)
```

Англоязычный синоним:

```
FindValue
```

Параметры:

<Дата>	Выражение, задающее значение даты, на которую требуется найти периодический реквизит справочника или периодическую константу.
<Режим>	Числовое выражение, значение которого задает режим поиска, если на заданную дату не существует значения периодического реквизита. Если -1 (минус единица) — возвращается значение на предыдущую дату, если 0 — возвращается код завершения неуспешной операции, если 1 — возвращается значение на последующую дату.

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод НайтиЗначение позволяет найти периодическое значение на заданную дату <Дата>. Режим поиска в случае, если на заданную дату не существует значения периодического реквизита задается параметром <Режим>. Само полученное значение следует считывать из атрибута Значение.

Пример:

```
Если ПерВал.НайтиЗначение(ДатаП, Нап) = 0 Тогда
   Предупреждение("Не нашли значение!", 3);
   Возврат;
КонецЕсли;
```

ВыбратьЗначения

Открыть выборку периодических значений по датам.

Синтаксис:

```
ВыбратьЗначения(<ДатаНачала>, <ДатаКонца>)
```

Англоязычный синоним:

```
Selectltems
```

Параметры:

<ДатаНачала>	Необязательный параметр. Выражение типа «дата» — дата начала периода выборки периодических значений. Если параметр не задан, то выборка начинается с самой ранней имеющейся даты.
<ДатаКонца>	Необязательный параметр. Выражение типа «дата» — дата конца периода выборки периодических значений. Если параметр не задан, то выборка заканчивается самой последней имеющейся датой.

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод ВыбратьЗначения предоставляет возможность (открывает выборку) выбирать периодические значения при помощи метода ПолучитьЗначение.

Дальнейшая выборка при помощи метода ПолучитьЗначение будет происходить среди периодических значений текущего объекта применения, заданного методом ИспользоватьОбъект.

Пример:

```
Процедура КурсыВалюты(КодВалюты)
   Вал = СоздатьОбъект("Справочник.Валюты");
   // Позиционируем созданный объект Вал по известному коду
```

```
    Вал.НайтиПоКоду(КодВалюты);
    Если Вал.Выбран() = 1 Тогда
       Доллар = Вал.ТекущийЭлемент();
    Иначе
       Предупреждение("Не найдена валюта!");
       Возврат;
    КонецЕсли;
    ПерВал = СоздатьОбъект("Периодический");
    ПерВал.ИспользоватьОбъект("Текуш_курс", Доллар);
    ПерВал.ОбратныйПорядок(1);
    ПерВал.ВыбратьЗначения();
    Пока ПерВал.ПолучитьЗначение() = 1 Цикл
       Сообщить("Курс: " + Строка(ПерВал.ДатаЗнач) + ПерВал.Значение);
    КонецЦикла;
КонецПроцедуры
```

См. также: ПолучитьЗначение, ИспользоватьОбъект

ВыбратьПоДокументу

Открыть выборку всех изменений периодических реквизитов справочника, сделанных документом.

Синтаксис:

```
ВыбратьПоДокументу(<Документ>)
```

Англоязычный синоним:

```
SelectByDoc
```

Параметры:

<Документ>	Выражение со значением типа «документ».

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод ВыбратьПоДокументу предоставляет возможность (открывает выборку) выбирать все периодические значения (по всем справочникам и реквизитам), заданные в справочнике документом.

Дальнейшая выборка осуществляется при помощи метода ПолучитьЗначение. При использовании данного метода не используется (игнорируется) установка объекта применения, задаваемая методом ИспользоватьОбъект.

См. также: УстановитьРеквизитСправочника, ТекущийОбъект, ТекущийРеквизит, ПолучитьЗначение

Пример:

```
Процедура ПечатьУстановокДокумента(Док)
    Если Док.Выбран() = 0 Тогда
       Возврат;
    КонецЕсли;
    Таб = СоздатьОбъект("Таблица");
    Таб.ИсходнаяТаблица("ПечатьИстории");
    Таб.ВывестиСекцию("Отчет" );
    Ист = СоздатьОбъект ("Периодический");
    Ист.ВыбратьПоДокументу(Док);
    Пока Ист.ПолучитьЗначение() = 1 Цикл
       Таб.ВывестиСекцию("Строка");
    КонецЦикла;
    Таб.Опции(0, 0, 0, 0);
    Таб.ТолькоПросмотр(1);
    Таб.Показать("Отчет");
КонецПроцедуры
```

ПолучитьЗначение

Получить из выборки следующее периодическое значение.

Синтаксис:

```
ПолучитьЗначение()
```

Англоязычный синоним:

```
GetValue
```

Возвращаемое значение:

Число: 1 — если элемент выбран успешно, 0 — если элемент не выбран (отсутствует).

Описание:

Метод ПолучитьЗначение выбирает периодическое значение в последовательности выборки, открытой перед этим при помощи метода ВыбратьЗначения или ВыбратьПоДокументу. Само полученное значение следует считывать из атрибута Значение. Дату этого значения можно считать из атрибута ДатаЗнач. Если значение периодического реквизита установлено документом, то этот документ можно получить при помощи метода ТекущийДокумент. Метод ТекущийОбъект позволит определить значение текущего элемента справочника, а метод ТекущийРеквизит — определить наименование текущего реквизита справочника.

Пример:

```
Процедура КурсыВалюты(КодВалюты)
   Вал = СоздатьОбъект("Справочник.Валюты");
   // Позиционируем созданный объект Вал по известному коду
   Вал.НайтиПоКоду(КодВалюты);
   Если Вал.Выбран() = 1 Тогда
      Доллар = Вал.ТекущийЭлемент();
   Иначе
      Предупреждение("Не найдена валюта!");
      Возврат;
   КонецЕсли;
   ПерВал = СоздатьОбъект("Периодический");
   ПерВал.ИспользоватьОбъект("Текущ_курс", Доллар);
   ПерВал.ОбратныйПорядок(1);
   ПерВал.ВыбратьЗначения();
   Пока ПерВал.ПолучитьЗначение() = 1 Цикл
      Сообщить("Курс на " + Строка(ПерВал.ДатаЗнач) + " = " + ПерВал.Значение);
   КонецЦикла;
КонецПроцедуры
```

См. также: ВыбратьЗначения, ВыбратьПоДокументу, ТекущийОбъект, ТекущийРеквизит, ТекущийДокумент

ОбратныйПорядок

Установить порядок выборки периодических значений.

Синтаксис:

```
ОбратныйПорядок(<Режим>)
```

Англоязычный синоним:

```
BackwardOrder
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение, значение которого задает режим выборки периодических значений. Если 0 — прямой порядок выборки, если значение отлично от 0 — обратный порядок выборки. По умолчанию — 1.

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод ОбратныйПорядок устанавливает порядок выборки периодических значений. Данный метод используется до вызова метода ВыбратьЗначения, который фактически открывает выборку. Дальнейшая выборка при помощи ПолучитьЗначение будет происходить в заданном порядке.

Пример:

```
Процедура ЗаполнитьСписокОбр()
   Вал = СоздатьОбъект("Справочник.Валюты");
   // Позиционируем созданный объект Вал по известному коду
   Вал.НайтиПоКоду(1);
   Если Вал.Выбран() = 1 Тогда
      Доллар = Вал.ТекущийЭлемент(};
   Иначе
      Предупреждение("Не найдена валюта!");
      Возврат;
   КонецЕсли;
   ПерВал = СоздатьОбъект("Периодический");
   ПерВал.ИспользоватьОбъект("Текущ_курс", Доллар);
   ПерВал.ОбратныйПорядок(1);
   ПерВал.ВыбратьЗначения();
   Пока ПерВал.ПолучитьЗначение() = 1 Цикл
      СпВал.ДобавитьЗначение(ПерВал.Значение, Строка(ПерВал.ДатаЗнач) +
                             ПерВал.Значение);
   КонецЦикла;
КонецПроцедуры
```

См. также: ВыбратьЗначения, ПолучитьЗначение

ТекущийДокумент

Возвращает документ, который установил значение периодического реквизита справочника.

Синтаксис:

```
ТекущийДокумент()
```

Англоязычный синоним:

```
Current Document
```

Возвращаемое значение:

Документа, который задал значение периодического реквизита справочника.

Описание:

Метод ТекущийДокумент возвращает документ, который задал значение периодического реквизита справочника. Метод используется после получения очередного значения из выборки (см. метод ПолучитьЗначение).

Пример:

```
Процедура ПечатьИстории(Элем)
   Если Элем.Выбран() = 0 Тогда
      Возврат;
   КонецЕсли;
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("ПечатьИстории");
   Таб.ВывестиСекцию("Отчет");
   Ист = СоздатьОбъект("Периодический");
   // Просмотрим все периодические реквизиты справочника
   Ист.ИспользоватьОбъект("", Элем);
   Ист.ВыбратьЗначения();
   Пока Ист.ПолучитьЗначение() = 1 Цикл
      ДокИст = Ист.ТекущийДокумент();
      Если ДокИст.Выбран() = 1 Тогда
         // если значение задано документом
         Таб.ВывестиСекцию("ПоДокументу");
      Иначе
         // если значение задано интерактивно
         Таб.ВывестиСекцию("Строка");
      КонецЕсли;
   КонецЦикла;
   Таб.Опции(0, 0, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("Отчет");
КонецПроцедуры
```

См. также: УстановитьРеквизитСправочника

ТекущийОбъект

Возвращает значение текущего элемента справочника.

Синтаксис:

```
ТекущийОбъект()
```

Англоязычный синоним:

```
CurrentObj
```

Возвращаемое значение:

Значение текущего элемента справочника.

Описание:

Метод ТекущийОбъект возвращает значение текущего элемента справочника. Метод используется после получения очередного значения из выборки, особенно полезен при выборке по документу (см. методы ВыбратьПоДокументу, ПолучитьЗначение).

Пример:

```
Процедура ПечатьУстановокДокумента(Док)
   // выведем все установки, которые произвел документ
   Если Док.Выбран() = 0 Тогда
      Возврат;
   КонецЕсли;
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("ПечатьИстории");
   Таб.ВывестиСекцию("Отчет");
   Ист = СоздатьОбъект("Периодический");
   Ист.ВыбратьПоДокумекту(Док);
   Пока Ист.ПолучитьЗначение() = 1 Цикл
      ИстОбъект = Ист.ТекущийОбъект();
      Если ИстОбъект.Вид() = "Товары" Тогда
         // если документом заданы изменения в справочнике Товары
         Таб.ВывестиСекцию("ПоТовару");
      ИначеЕсли ИстОбъект.Вид() = "Клиенты" Тогда
         // если документом заданы изменения в справочнике Клиенты
         Таб.ВывестиСекцию("ПоКлиенту");
      КонецЕсли;
   КонецЦикла;
   Таб.Опции(0, 0, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("Отчет");
КонецПроцедуры
```

См. также: УстановитьРеквизитСправочника, ВыбратьПоДокументу, ПолучитьЗначение

ТекущийРеквизит

Возвращает наименование текущего реквизита справочника.

Синтаксис:

ТекущийРеквизит()

Англоязычный синоним:

CurrentAttrubute

Возвращаемое значение:

Строковое значение — наименование текущего реквизита справочника.

Описание:

Метод ТекущийРеквизит возвращает наименование текущего реквизита справочника. Если выборка производится по документу, то в возвращаемом значении сначала записано наименование справочника, а затем через точку наименования реквизита. Метод используется после получения очередного значения из выборки, особенно полезен при выборке по документу или по всем реквизитам.

Пример:

```
Процедура ПечатьУстановокДокумента(Док)
   // выведем все установки, которые произвел документ
   Если Док.Выбран() = 0 Тогда
      Возврат;
   КонецЕсли;
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("ПечатьИстории");
   Таб.ВывестиСекцию("Отчет");
   Ист = СоздатьОбъект("Периодический");
   Ист.ВыбратьПоДокументу(Док);
   Пока Ист.ПолучитьЗначение() = 1 Цикл
      Если Ист.ТекущийРеквизит() = "Товары.Цена" Тогда
         // если документом заданы изменения Цены Товара
         Таб.ВывестиСекцию("ПоЦенеТовара");
      ИначеЕсли Ист.ТекущийРеквизит() = "Клиенты.Статус" Тогда
         // если документом заданы изменения Статуса Клиента
         Таб.ВывестиСекцию("ПоСтатусу Клиента");
      КонецЕсли;
   КонецЦикла;
   Таб.Опции(0, 0, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("Отчет");
КонецПроцедуры
```

См. также: УстановитьРеквизитСправочника, ВыбратьЗначения, ВыбратьПоДокументу, ПолучитьЗначение

НомерСтроки

Возвращает номер строки текущего документа, изменившего периодическое значение реквизита справочника.

Синтаксис:

НомерСтроки()

Англоязычный синоним:

LineNurn

Возвращаемое значение:

Числовое значение — номер строки документа, изменившего периодическое значение реквизита справочника.

Описание:

Метод НомерСтроки возвращает значение номера строки документа, изменившего периодическое значение реквизита справочника (в случае, когда в Модуле документа использовали метод ПривязыватьСтроку). Метод используется после получения очередного значения из выборки, особенно полезен при выборке по документу или по всем реквизитам (см. методы ВыбратьЗначения, ВыбратьПоДокументу, ПолучитьЗначение).

Пример:

```
Процедура ПечатьИстории(Элем)
   Если Элем.Выбран() = 0 Тогда
      Возврат;
   КонецЕсли;
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("ПечатьИстории");
   Таб.ВывестиСекцию("Отчет");
   Ист = СоздатьОбъект("Периодический");
   // Просмотрим все периодические реквизиты справочника
   Ист.ИспользоватьОбъект("", Элем);
   Ист.ВыбратьЗначения();
```

```
    Пока Ист.ПолучитьЗначение() = 1 Цикл
       ДокИст = Ист.ТекущийДокумент();
       Если ДокИст.Выбран() = 1 Тогда
          // если значение задано документом
          НомСтрДок = Ист.НомерСтроки();
          Таб.ВывестиСекцию("ПоДокументу");
       Иначе
          // если значение задано интерактивно
          Таб.ВывестиСекцию("Строка");
       КонецЕсли;
    КонецЦикла;
    Таб.Опции(0, 0, 0, 0);
    Таб.ТолькоПросмотр(1);
    Таб.Показать("Отчет");
КонецПроцедуры
```

См. также: УстановитьРеквизитСправочника, ВыбратьПоДокументу, ПолучитьЗначение

Записать

Записать (обновить) периодическое значение.

Синтаксис:

```
Записать()
```

Англоязычный синоним:

```
Write
```

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод Записать выполняет запись периодического значения на заданную дату. Если на заданную дату уже существует запись, то она модифицируется. До вызова данного метода следует само значение записать в атрибут Значение. Дату этого значения следует записать в атрибут ДатаЗнач.

Пример:

```
Процедура ЗаписьКурсаДоллара(Курс)
   Вал = СоздатьОбъект("Справочник.Валюты");
   // Позиционируем созданный объект Вал по известному коду
   Вал.НайтиПоКоду(1);
   Если Вал.Выбран() = 1 Тогда
      Доллар = Вал.ТекущийЭлемент();
   Иначе
      Предупреждение("Не найдена валюта!");
      Возврат;
   КонецЕсли;
   ПерВал = СоздатьОбъект("Периодический");
   ПерВал.ИспользоватьОбъект("Текущ_курс".Доллар);
   ПерВал.Значение = Курс;
   ПерВал.ДатаЗнач = РабочаяДата();
   ПерВал.Записать();
КонецПроцедуры
```

Удалить

Удалить периодическое значение.

Синтаксис:

```
Удалить()
```

Англоязычный синоним:

```
Delete
```

Возвращаемое значение:

Число: 1 — если вызов метода закончился успешно, 0 — если нет.

Описание:

Метод Удалить удаляет периодическое значение на заданную дату. До вызова данного метода атрибут ДатаЗнач должен содержать дату удаляемого периодического значения..

Пример:

```
Процедура ЧисткаКурсов()
   Вал = СоздатьОбъект("Справочник.Валюты");
   // Позиционируем созданный объект Вал по известному коду
   Вал.НайтиПоКоду(1);
   Если Вал.Выбран() = 1 Тогда
      Доллар = Вал.ТекущийЭлемент();
   Иначе
      Предупреждение("Не найдена валюта!");
      Возврат;
```

```
    КонецЕсли;
    ПерВал = СоздатьОбъект("Периодический");
    ПерВал.ИспользоватьОбъект("Текущ_курс".Доллар);
    ПерВал.ВыбратьЗначения();
    Пока ПерВал.ПолучитьЗначение() = 1 Цикл
       Курс = ПерВал.Значение;
       ДатаКурса = ПерВал.ДатаЗнач;
       Если Курс < 1000 Тогда
          ПерВал.Удалить();
       КонецЕсли;
    КонецЦикла;
КонецПроцедуры
```

Глава 28
Работа со Списком Значений

При формировании пользовательского интерфейса прикладной задачи довольно часто возникают ситуации, когда для удобства работы пользователя необходимо дать ему возможность выбирать какое-либо значение из заранее подготовленного списка. Возможности работы со справочниками, журналами и перечислениями предоставляют достаточно мощные средства выбора этих объектов. Однако, существует также необходимость в выборе значений из списков, сформированных и отсортированных нестандарнным образом, которые невозможно или неудобно получать непосредственно при работе со стандартными визуальными представлениями журналов, справочников или перечислений. Для этого в системе 1С:Предприятие используется специальный агрегатный тип данных — «СписокЗначений» и специальные элементы форм диалога: «Список» и «Поле со списком».

Объект типа «СписокЗначений» — это средство языка (не сохраняемый в БД объект), которое позволяет строить «динамические массивы» и манипулировать ими (добавлять, редактировать, удалять элементы, сортировать). Список значений может быть наполнен значениями любого типа, т. е. в одном списке типы хранимых значений могут быть разными. Одним из примеров использования данного объекта может служить организация выбора конкретного документа из списка возможных документов, сформированного по сложному алгоритму.

Контекст работы со Списком Значений

Во всех программных модулях вызов соответствующих методов может выполняться при помощи переменной со ссылкой на объект типа «СписокЗначений». Такие объекты можно создавать при помощи функции СоздатьОбъект, ссылка на который присваивается переменной. Кроме того, если в форму диалога при помощи визуальных средств конфигуратора вставлены специальные элементы форм диалога «Список» и «Поле со списком», то идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа «СписокЗначений».

Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после имени объекта.

При создании объекта типа «СписокЗначений» при помощи функции `СоздатьОбъект`, в качестве названия агрегатного типа данных обязательно Должно выступать ключевое слово `СписокЗначений`.

Англоязычный синоним ключевого слова `СписокЗначений` — `ValueList`.

Пример:

```
МойСписок = СоздатьОбъект("СписокЗначений");
```

Методы объекта Список Значений

ДобавитьЗначение

Добавить значение в список.

Синтаксис:

```
ДобавитьЗначение(<Значение>, <Строка>)
```

Англоязычный синоним:

```
AddValue
```

Параметры:

`<Значение>`	Выражение со значением, которое добавляется в список.
`<Строка>`	Необязательный параметр. Строковое выражение, содержащее задаваемое символьное представление добавляемого значения. По умолчанию принимает стандартное в системе 1С:Предприятие символьное представление объекта.

Описание:

Метод `ДобавитьЗначение` добавляет значение и его символьное представление в конец списка. Представление используется в дальнейшем для удобства сортировки и выбора значений из списка (отображается в диалоговом окне выбора).

Пример:

```
// Объявляем переменные
Перем Список;
Перем ВР;
// процедура выбора значения
Процедура ВыбратьВР()
   Если Список.ВыбратьЗначение(ВР, "Выбор дог: паты") = 1 Тогда
      Расчет = ВР;
      НазваниеВР = Расчет.Код;
   КонецЕсли;
КонецПроцедуры
// инициализация переменных
Список = СоздатьОбъект("СписокЗначений");
Список.ДобавитьЗначение(ВидРасчета.ДоплатаЛроцентом, "Процентом");
Список.ДобавитьЗначение(ВидРасчета.ДоплатаСуммой, "Суммой");
// первоначальное значение
НазваниеВР = Расчет.Код;
```

ВставитьЗначение

Вставить значение в указанную позицию списка.

Синтаксис:

```
ВставитьЗначение(<Позиция>, <Знач>, <Строка>, <Колич>)
```

АНГЛОЯЗЫЧНЫЙ СИНОНИМ:

```
InsertValue
```

Параметры:

<Позиция>	Числовое выражение — номер позиции в списке, начиная с которого будут вставлены новые значения. Номер позиции может быть от 1 до РазмерСписка() + 1.
<Знач>	Выражение со значением, которое добавляется в список.
<Строка>	Необязательный параметр. Строковое выражение, содержащее задаваемое символьное представление добавляемого значения. По умолчанию принимает стандартное в системе 1С:Предприятие символьное представление объекта.
<Колич>	Необязательный параметр. Числовое выражение — количество повторов. По умолчанию — 1.

Описание:

Метод ВставитьЗначение добавляет значение и его символьное представление в указанную позицию списка <Позиция> заданное число раз <Колич>. Представление используется в дальнейшем для удобства сортировки и выбора значений из списка (отображается в диалоговом окне выбора).

Пример:

```
// Объявляем переменные
Перем Список;
Перем ВР;
// процедура выбора значения
Процедура ВыбратьВР()
   Если Список.ВыбратьЗначение(ВР, "Выбор доплаты") = 1 Тогда
      Расчет = ВР;
      НазваниеВР = Расчет.Код;
   КонецЕсли;
КонецПроцедуры
// инициализация переменных
Список = СоздатьОбъект("СписокЗначений");
Список.ВставитьЗначение(1, Вид Расчета.ДоплатаПроцентом, "Процентом");
Список.ВставитьЗначение(1, Вид Расчета.ДоплатаСуммой, "Суммой");
// первоначальное значение
НазваниеВР = Расчет.Код;
```

РазмерСписка

Определить размер списка.

Синтаксис:

```
РазмерСписка()
```

Англоязычный синоним:

```
GetListSize
```

Возвращаемое значение:

Числовое значение — количество элементов в списке.

Описание:

Метод РазмерСписка позволяет определить общее количество элементов в списке.

Пример:

```
Разм = Список.РазмерСписка();
```

НайтиЗначение

Определить номер позиции в списке для элемента, имеющего заданное значение.

Синтаксис:

```
НайтиЗначение(<Знач>)
```

Англоязычный синоним:

```
FindValue
```

Параметры:

<Знач>	Выражение со значением, которое необходимо найти в списке.

Возвращаемое значение:

Номер позиции в списке, где расположено требуемое значение. Если значение не найдено, то — 0.

Описание:

С помощью метода НайтиЗначение можно определить номер позиции в списке для элемента, имеющего значение <Знач>.

Пример:

```
Позиция = Спис.НайтиЗначение(Вид Расчета.ДоплатаПроцентом);
```

ПолучитьЗначение

Получить значение элемента по номеру в списке.

Синтаксис:

```
ПолучитьЗначение(<Позиция>, <Перем>)
```

Англоязычный синоним:

```
GetValue
```

Параметры:

<Позиция>	Числовое выражение — номер элемента в списке, значение которого будет возвращено. Номер позиции может быть от 1 до количества элементов в списке.
<Перем>	Идентификатор переменной в которую будет возвращено строковое выражение, содержащее символьное представление получаемого значения.

Возвращаемое значение:

Полученное значение из списка.

Описание:

С помощью метода ПолучитьЗначение можно получить значение элемента, находящегося в указанной позиции списка.

Пример:

```
ОпредПредст = "";
ОпредВид=Спис.ПолучитьЗначение(1, ОпредПредст);
```

УстановитьЗначение

Установить значение в указанной позиции списка.

Синтаксис:

```
УстановитьЗначение(<Позиция>, <Знач>, <Строка>, <Колич>)
```

Англоязычный синоним:

```
SetValue
```

Параметры:

<Позиция>	Числовое выражение — номер позиции в списке, начиная с которого будут установлены новые значения, т. е. «старые» будут заменены. Номер позиции может быть от 1 до РазмерСписка+1.
<Знач>	Выражение со значением, которое добавляется в список.
<Строка>	Необязательный параметр. Строковое выражение, содержащее задаваемое символьное представление добавляемого значения. По умолчанию принимает стандартное в системе 1С:Предприятие символьное представление объекта.
<Колич>	Необязательный параметр. Числовое выражение — количество повторов. По умолчанию — 1.

Описание:

Метод УстановитьЗначение устанавливает значения и его символьные лредставления, начиная с указанной позиции списка <Позиция> заданное число повторов <Колич>. Представление используется в дальнейшем для удобства сортировки и выбора значений из списка (отображается в диалоговом окне выбора).

Пример:

```
Спис.УстановитьЗначение(1, ВидРасчета.ДоплатаПроцентом, "Процентом");
```

Получить

Получить значение элемента по указанному представлению.

Синтаксис:

```
Получить(<Строка>)
```

Англоязычный синоним:

```
Get
```

Параметры:

<Строка>	Строковое выражение, содержащее задаваемое символьное представление получаемого значения.

Возвращаемое значение:

Полученное значение из списка.

Описание:

Метод Получить возвращает значение по указанному представлению. Если значения с таким представлением нет, то возвращается пустое значение.

Пример:

```
ОпредЗнач = Спис.Получить("ОпредПредст");
```

Установить

Установить значение с указанным представлением.

Синтаксис:

```
Установить(<Строка>, <Знач>)
```

Англоязычный синоним:

```
Set
```

Параметры:

<Строка>	Строковое выражение, содержащее задаваемое символьное представление добавляемого значения.

<Знач> Выражение со значением, которое добавляется в список.

Описание:

Метод Установить устанавливает в списке значение с указанным представлением, если значение с таким представлением уже есть — изменятся значение, если нет — добавляется в конец списка значение с указанным представлением.

Пример:

```
Спис.Установить("Процентом", ВидРасчета.ДоплатаПроцентом);
```

УдалитьЗначение

Удалить значение в указанной позиции списка.

Синтаксис:

```
УдалитьЗначение(<Позиция>, <Колич>)
```

Англоязычный синоним:

```
RemoveValue
```

Параметры:

<Позиция> Числовое выражение — номер позиции в списке, начиная с которой будут удалены значения. Номер позиции может быть от 1 до количества элементов в списке.

<Колич> Необязательный параметр. Числовое выражение — количество повторов. По умолчанию — 1.

Описание:

Метод УдалитьЗначение удаляет <Колич> значений, начиная с указаний позиции <Позиция>.

Пример:

```
Список.УдалитьЗначение(1, 2);
```

УдалитьВсе

Удалить все элементы списка.

Синтаксис:

```
УдалитьВсе()
```

Англоязычный синоним:

```
RemoveAll
```

Описание:

Метод УдалитьВсе удаляет все элементы списка.

Пример:

```
Список.УдалитьВсе();
```

Сортировать

Отсортировать список по значениям.

Синтаксис:

```
Сортировать(<Направление>, <ДокументыПоДате>)
```

Англоязычный синоним:

```
Sort
```

Параметры:

<Направление> Необязательный параметр. Числовое выражение. Если — 0, то сортировка по возрастанию. Если не 0 то сортировка по убыванию. Значение по умолчанию — 0.

<ДокументыПоДате> Необязательный параметр. Имеет смысл только в том случае, если значениями списка значений являются документы. В этом случае можно задавать сортировку документов по их хронологии. Число: 1 — сортировка по хронологии документов; 0 — нет. Значение по умолчанию — 0.

Описание:

Метод Сортировать выполняет сортировку списка по значениям. Если первый элемент списка (значение в первой позиции) имеет базовый тип данных (число, строка, дата), то сортировка проводится по значениям элементов, приведенных стандартными правилами к типу первого элемента. В противном случае, если первый элемент имеет агрегатный тип данных, сортировка проводится по значениям элементов, приведенных к строковому представлению типа данных.

Пример:

```
Спис.Сортировать();
```

СортироватьПоПредставлению

Отсортировать список по представлению.

Синтаксис:

```
СортироватьПоПредставлению(<Направление>)
```

Англоязычный синоним:

```
SortByPresent
```

Параметры:

<Направление> Необязательный параметр. Числовое выражение. Если — 0, то сортировка производится по возрастанию. Если отлично от 0, то сортировка по убыванию.

Описание:

Метод СортироватьПоПредставлению выполняет сортировку списка по символьному представлению в соответствии с порядком, заданным параметром <Направление>.

Пример:

```
Спис.СортироватьПоПредставлению();
```

СдвинутьЗначение

Переместить значение списка значений на новую позицию.

Синтаксис:

```
СдвинутьЗначение(<Колич>, <НомерПоз>)
```

Англоязычный синоним:

```
MoveValue
```

Параметры:

<Колич>	Числовое выражение — количество позиций, на которое надо переместить значение. Если число положительное, то значение сдвигается вниз, если отрицательное, то вверх.
<НомерПоз>	Номер позиции, значение которой надо переместить.

Описание:

С помощью метода СдвинутьЗначение можно переместить значение списка значений на новую позицию.

Пример:

```
Спис.СдвинутьЗначение(1, Поз);
```

Принадлежит

Проверяет вхождение в список значений указанного значения.

Синтаксис:

```
Принадлежит(<Значение>)
```

Англоязычный синоним:

```
Belong
```

Параметры:

<Значение>	Значение, проверяемое на вхождение в список значений.

Возвращаемое значение:

Число: 1 — если проверяемое значение входит в список значений; 0 — если не входит.

Описание:

Метод Принадлежит проверяет вхождение в список значений заданного значения. Другими словами, оператор Список.Принадлежит(ВыбДок) проверяет, является ли значение ВыбДок подмножеством списка значений Список.

Если на принадлежность проверяется значение типа элемент справочника, то проверка выполняется с учетом его возможного вхождения в группы справочников, которые являются значениями списка значений.

Данный метод оптимизирует проверку принадлежности при массовых последовательных сравнениях, если между сравнениями сам список значений не меняется.

Пример:

```
Процедура ПроверитьКлиентов(СписокГруппКлиентов)
   // в качестве параметра Процедуры передается СписокЗначений
   // в котором записаны как элементы справочника Клиенты, так и
   // группы этого справочника
   Док = СоздатьОбъект("Документ.Накладная");
   Док.ВыбратьДокументы();
   Пока Док.ПолучитьДокумент() = 1 Цикл
      Если СписокГруппКлиентов.Принадлежит(Док.Клиент) = 1 Тогда
         Сообщить(Док.Клиент.Наименование + " " + Док);
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

ВыбратьЗначение

Открыть окно для интерактивного выбора значения из списка.

Синтаксис:

```
ВыбратьЗначение(<Значение>, <Заголовок>, <Позиция> , <Таймаут>, <Способ6Выбора>)
```

Англоязычный синоним:

```
ChooseValue
```

Параметры:

<Значение>	Идентификатор переменной, куда помещается значение — результат выбора.
<Заголовок>	Строковое выражение, значение которого отображается в заголовке диалогового окна. Может использоваться для подсказки пользователю.
<Позиция>	Идентификатор переменной, куда помещается номер позиции выбранного значения в списке.
<Таймаут>	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя.
<СпособВыбора>	Необязательный параметр. Числовое выражение, значение которого задает способ выбора

значения:
0 — в виде диалога;
1 — выбор производится в виде меню, которое подстраивается по месту текущего элемента диалога или ячейки таблицы, откуда вызвано;
2 — выбор маленьким списком (список похож на выбор значения перечисления), также привязанным к позиции элемента диалога.
Значение по умолчанию — 0.

Возвращаемое значение:

Число: 1 — если выбор произведен (нажата кнопка «ОК»); 0 — если выбор не призведен (нажата кнопка «ОТМЕНА»); -1 (минус единица) — закончилось время <Таймаут> ожидания отклика пользователя.

Описание:

Метод ВыбратьЗначение вызывает диалоговое окно для интерактивного выбора значения из заранее подготовленного списка. Возвращаемое числовое значение: 1 — если выбор произведен, 0 — если нет. Параметр <Значение> указывает на значение того элемента списка, на котором будет установлен курсор при открытии окна выбора. В этот же параметр возвращается выбранное значение. В параметр <Позиция> возвращается номер позиции выбранного значения в списке. Если выбор отменен, то значение параметров не меняются.

Данный метод может использоваться только для переменных созданных функцией СоздатьОбъект.

Пример:

```
Процедура ВыбратьКредит(ДокКредита)
   Список = СоздатьОбъект("СписокЗначений");
   Рег = СоздатьОбъект("Регистр.ТоварныйКредит");
   Рег.УстановитьФильтр(Клиент, );
   Рег.ВыбратьИтоги();
   Пока Рег.ПолучитьИтог() = 1 Цикл
      Док = Рег.Документ;
      Список.ДобавитьЗначение(Док, "" + Док + " — Остаток = " + Рег.Долг);
   КонецЦикла;
   Если Список.ВыбратьЗначение(ВыбДок, "Выберите кредит") = 1 Тогда
      ДокКредита = ВыбДок;
   КонецЕсли;
КонецПроцедуры
```

ОтметитьЗначения

Открыть окно для интерактивной пометки значений списка.

Синтаксис:

ОтметитьЗначения(<Значение>, <Заголовок>, <Позиция>, <Таймаут>)

Англоязычный синоним:

CheckValues

Параметры:

<Значение>	Идентификатор переменной, куда помещается значение — результат выбора.
<Заголовок>	Строковое выражение, значение которого отображается в заголовке диалогового окна. Может использоваться для подсказки пользователю.
<Позиция>	Необязательный параметр. Идентификатор переменной, куда помещается номер позиции выбранного значения в списке.
<Таймаут>	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя.

Возвращаемое значение:

Число: 1 — если выбор произведен (нажата кнопка «ОК»); 0 — если выбоо не произведен (нажата кнопка «ОТМЕНА»); -1 (минус единица) — закончилось время <Таймаут> ожидания отклика пользователя.

Описание:

Метод ОтметитьЗначения вызывает диалоговое окно для интерактивной пометки значений списка из заранее подготовленного списка.

Возможность выбора значения — это дополнительная возможность, предоставляемая данным методом. Параметр <Значение> указывает на значение того элемента списка, на котором будет установлен курсор при открытии окна выбора. В этот же параметр возвращается последнее выбранное значение- В параметр <Позиция> возвращается номер позиции последнего выбранного значения в списке. Если выбор отменен, то значение параметров не меняются.

Данный метод может использоваться только для переменных созданных функцией СоздатьОбъект.

Пример:

```
Функция ВыбратьКредит (ДокКредита)
   Перем ВыбДок;
   Список = СоздатьОбъект("СписокЗначений");
   Рег = СоздатьОбъект("Регистр.ТоварныйКредит");
   Рег.УстановитьФильтр(Клиент, );
   Рег.ВыбратьИтоги();
   Пока Рег.ПолучитьИтог() = 1 Цикл
      Док = Рег.Документ;
      Список. ДобавитьЗначение(Док, "" + Док + " — Остаток = " + Рег.Долг);
```

```
   КонецЦикла;
   Список.ОтметитьЗначения(ВыбДок, "Выберите кредиты");
   Возврат Список;
КонецФункции
```

Пометка

Пометить значение списка.

Синтаксис:

`Пометка(<Позиция>, <Отметка>}`

Англоязычный синоним:

`Check`

Параметры:

`<Позиция>` Номер позиции выбранного значения в списке.

`<Отметка>` Необязательный параметр. Число: 1 — установить отметку; 0 — снять отметку. Если данный параметр опущен, то отметка значения списка не изменяется.

Возвращаемое значение:

Значение метки до выполнения метода. Число: 1 — отметка установлена; 0 _- отметка не установлена.

Описание:

Метод `Пометка` позволяет пометить указанную позицию списка значений.

Пример:

```
Функция ОтметитьДокумент(Список, ВыбДок)
   Для Н = 1 по Список.РазмерСписка() Цикл
      Док = Список.ПолучитьЗначение(Н);
      Если Док = ВыбДок Тогда
         Список.Пометка(Н, 1);
      КонецЕсли;
   КонецЦикла;
КонецФункции
```

ТекущаяСтрока

Установить/определить текущий элемент списка в элементе диалога типа «Список» или «Поле со списком».

Синтаксис:

`ТекущаяСтрока(<ИндексСтроки>)`

Англоязычный синоним:

`CurSel`

Параметры:

`<ИндексСтроки>` Необязательный параметр. Числовое выражение с задаваемым индексом строки для элемента диалога типа «Список» или «Поле со списком», на которое требуется установить курсор. Если параметр не задан, то положение курсора в поле диалога не меняется.

Возвращаемое значение:

Числовое значение, соответствующее индексу текущей строки поля диалога (до его изменения).

Описание:

Метод `ТекущаяСтрока` в тексте программного модуля можно использовать синтаксически как функцию или процедуру. Данный метод позволяет установить и/или считать текущее положение курсора в элементе диалога типа «Список» или «Поле со списком».

Данный метод можно использовать только для объектов, которые созданы при помощи визуальных средств конфигуратора (в форму вставлены элементы Диалога типа «Список» или «Поле со списком», а идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа «СписокЗначений»).

Пример:

```
Список.ТекущаяСтрока(2);
```

ИзСтрокиСРазделителями

Заполнить список значениями из переданной строки.

Синтаксис:

`ИзСтрокиСРазделителями(<Строка>)`

Англоязычный синоним:

`FromSeparatedString`

Параметры:

`<Строка>` Строковое выражение, содержащее строки в двойных кавычках и числа, разделенные запятыми.

Описание:

Метод `ИзСтрокиСРазделителями` заполняет список значениями из переданной в качестве параметра строки, содержащей строки в двойных кавычках и числа, разделенные запятыми.

Пример:

```
Спис.ИзСтрокиСРазделителями("5, 6, 12, 68, ""ОпредПредст""");
```

ВСтрокуСРазделителями

Преобразует список значений в строку, содержащую строки в двойных кавычках и числа, разделенные запятыми.

Синтаксис:

```
ВСтрокуСРазделителями()
```

Англоязычный синоним:

```
ToSeparatedString
```

Описание:

Метод `ВСтрокуСРазделителями` преобразует список значений в строку, содержащую строки в двойных кавычках и числа, разделенные запятыми.

Пример:

```
СтрПроводка = Спис.ВСтрокуСРазделителями();
```

Выгрузить

Выгрузить список значений в другой список значений или таблицу значений.

Синтаксис:

```
Выгрузить(<Знач>, <НачПоз>, <Колич>)
```

Англоязычный синоним:

```
Unload
```

Параметры:

`<Знач>`	Значение типа «Таблица значений» или «Список значений», в которое нужно выгрузить данные. При выгрузке в таблицу значений, в ней добавляется новая колонка. Если переданное значение пустое, тогда система сама создаст объект типа «Список значений».
`<НачПоз>`	Необязательный параметр. Номер начальной позиции, с которой надо начинать выгрузку. Значение по умолчанию 1.
`<Колич>`	Необязательный параметр. Количество выгружаемых значений, если не указан то все.

Описание:

Метод `Выгрузить` позволяет выгрузить список значений в другой список значений или таблицу значений. При выгрузке в таблицу значений, в ней добавляется новая колонка.

Пример:

```
СтарСписок.Выгрузить(НовСписок);
```

Глава 29
Работа с Таблицей Значений

Объект типа «ТаблицаЗначений» ~ это средство языка (не сохраняемый в информационной базе объект), которое позволяет строить «динамические массивы» и манипулировать ими (добавлять, редактировать, удалять элементы, сортировать). Таблица значений может быть наполнена значениями любого типа, т. е. в одной таблице типы хранимых значений могут быть разными. Таблица значений может использоваться и как простой двумерный массив (матрица) значений и как таблица строк определенной структуры, когда колонки типизованы и имеют идентификаторы.

Специальный элемент формы диалога «ТаблицаЗначений» является интерфейсным средством отображения и манипулирования этим объектом в формах диалогов. При формировании пользовательского интерфейса прикладной задачи довольно часто возникают ситуации, когда для удобства работы пользователя необходимо дать ему возможность выбирать какое-либо значение из заранее подготовленной таблицы значений. Возможности работы со справочниками, журналами и перечислениями предоставляют достаточно мощные средства выбора этих объектов. Однако, существует также необходимость в выборе значений из таблиц, сформированных и отсортированных нестандартным образом, которые невозможно или неудобно получать непосредственно при работе со стандартными визуальными представлениями журналов, справочников или перечислений.

Одним из примеров использования данного объекта может служить организация выбора конкретного товара из таблицы возможных товаров, сформированной по сложному алгоритму.

При работе с таблицей значений следует различать два понятия текущей строки таблицы. Во-первых, существует текущая строка объекта «ТаблицаЗначений», которая устанавливается при помощи методов `ПолучитьСтрокуПоНомеру`, `ВыбратьСтроки`, `ПолучитьСтроку`, эта текущая строка меняется программно и служит для обращения к значениям колонок с помощью идентификаторов уже без указания строки. Во-вторых, существует текущая строка элемента формы диалога «ТаблицаЗначений», которая является активной строкой (на ней установлен курсор) в диалоге. Текущая строка элемента диалога возвращается и устанавливается исключительно только при помощи метода `ТекущаяСтрока`. Например, если программно сменить текущую строку объекта, курсор в визуальном представлении диалога сам собой не передвинется, это можно выполнить только при помощи метода `ТекущаяСтрока`. Однако, при вызове из формы процедуры обработки, текущая строка объекта будет совпадать с текущей строкой визуального представления.

Контекст работы с Таблицей Значений

Во всех программных модулях вызов соответствующих методов может выполняться при помощи переменной со ссылкой на объект типа «ТаблицаЗначений». Такой объект можно создавать при помощи функции `СоздатьОбъект` ссылка на который присваивается переменной. Кроме того, если в форму диалога при помощи визуальных средств конфигуратора вставлен специальный элемент формы диалога «ТаблицаЗначений», то идентификатор этого элемента доступен в контексте программного модуля этой формы как уже существующий объект типа «ТаблицаЗначений».

Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после имени объекта.

При создании объекта типа «ТаблицаЗначений» при помощи функции `СоздатьОбъект`, в качестве названия агрегатного типа данных обязательно должно выступать ключевое слово `ТаблицаЗначений`.

Англоязычный синоним ключевого слова `ТаблицаЗначений` — `ValueTable`.

Пример:

```
МояТаблицаЗначений = СоздатьОбъект("ТаблицаЗначений");
```

Атрибуты Таблицы Значений

НомерСтроки

Номер текущей строки таблицы.

Синтаксис:

`НомерСтроки`

Англоязычный синоним:

`LineNo`

Описание;

Атрибут (только для чтения) `НомерСтроки` содержит номер текущей строки таблицы значений.

<ИдентификаторКолонки>

Значение элемента таблицы в текущей строке в колонке, заданной идентификатором.

Синтаксис:

`<ИдетификаторКолонки>`

Описание:

Данный атрибут содержит значение таблицы значений в текущей строке в колонке, заданной идентификатором. В тексте программного модуля используется идентификатор конкретной колонки, который задается при создании колонки при помощи методов `ДобавитьКолонку` и `ВставитьКолонку`. Если при создании колонки идентификатор колонки не указан, то обращение к колонке возможно только по номеру.

Методы объекта Таблица Значений

КоличествоКолонок

Устанавливает/возвращает количество колонок.

Синтаксис:

```
КоличествоКолонок(<КоличКолонок>)
```

Англоязычный синоним:

```
ColumnCount
```

Параметры:

<КоличКолонок>	Необязательный параметр. Числовое выражение, которое указывает новое количество колонок. Если параметр опущен, то количество колонок не меняется.

Возвращаемое значение: количество колонок до вызова метода.

Описание:

Метод КоличествоКолонок устанавливает/возвращает количество колонок таблицы значений.

Пример:

```
Разм = ТаблицаТоваров.КоличествоКолонок();
```

НоваяКолонка

Добавить в конец таблицы значений новую колонку.

Синтаксис:

```
НоваяКолонка(<Идентификатор>, <Тип>, <Длина>, <Точность>, <Заголовок>, <Ширина>,
             <Формат>, <Положение>)
```

Англоязычный синоним:

```
NewColumn
```

Параметры:

<Идентификатор>	Необязательный параметр. Идентификатор колонки, если не указан обращение к колонке возможно только по номеру.
<Тип>	Необязательный параметр. Строка или вид субкон-то, задающий тип колонки. Если не указан, то можно хранить любой тип.
<Длина>	Необязательный параметр. Длина для числовой или строковой колонки.
<Точность>	Необязательный параметр. Точность (длина дробной части) для числовой колонки.
<Заголовок>	Необязательный параметр. Строковое выражение содержащее заголовок колонки в элементе диалога типа «ТаблицаЗначений».
<Ширина>	Необязательный параметр. Числовое выражение, содержащее ширину колонки (в символах) для представления колонки в элементе диалога типа «ТаблицаЗначений».
<Формат>	Необязательный параметр. Строковое выражение, содержащее форматную строку, которая будет использована при визуальном отображении значений данной колонки.
<Положение>	Необязательный параметр. Определяет вариант выравнивания при визуальном отображении значений данной колонки. Число: 1 — слева; 2 — справа.

Возвращаемое значение: номер новой колонки.

Описание:

Метод НоваяКолонка добавляет в конец таблицы значений новую колонку,

Пример:

```
Табл.НоваяКолонка("Код", "Число", 6, 0, "Код", 6);
```

ВставитьКолонку

Вставить новую колонку в таблицу значений в указанную позицию.

Синтаксис:

```
ВставитьКолонку(<Идентификатор>, <НомерКолонки>, <Тип>, <Длина>, <Точность>,
                <Заголовок>, <Ширина>, <Формат>, <Положение>)
```

Англоязычный синоним:

```
InsertColumn
```

Параметры:

<Идентификатор>	Необязательный параметр. Идентификатор колонки, если не указан обращение к колонке возможно только по номеру.
<НомерКолонки>	Необязательный параметр. Числовое выражение, содержащее позицию, в которую вставляется новая колонка.
<Тип>	Необязательный параметр. Строка или вид суб кон-то, задающий тип колонки. Если не указан, то можно хранить любой тип.
<Длина>	Необязательный параметр. Длина для числовой или строковой колонки.
<Точность>	Необязательный параметр. Точность (длина дробной части) для числовой колонки.
<Заголовок>	Необязательный параметр. Строковое выражение, содержащее заголовок колонки в элементе диалога типа «ТаблицаЗначений».
<Ширина>	Необязательный параметр. Числовое выражение, содержащее ширину колонки (в символах) для представления колонки в элементе диалога типа «ТаблицаЗначений».
<Формат>	Необязательный параметр. Строковое выражение, содержащее форматную строку, которая будет использована при визуальном отображении значений данной колонки.

<Положение>	Необязательный параметр. Определяет вариант выравнивания при визуальном отображении значений данной колонки. Число: 1 — слева; 2 — справа.

Возвращаемое значение: номер новой колонки.

Описание:

Метод ВставитьКолонку вставляет новую колонку в таблицу значений в указанную позицию.

Пример:

```
Табл.ВставитьКолонку("Код", 1, "Число", 6, 0, "Код", 6);
```

УдалитьКолонку

Удаляет колонку из таблицы значений.

Синтаксис:

```
УдалитьКолонку(<Колонка>)
```

Англоязычный синоним:

```
DeleteColumn
```

Параметры:

<Колонка>	Номер или идентификатор колонки.

Описание:

Метод УдалитьКолонку удаляет колонку <Колонка> из таблицы значений.

Пример:

```
ТаблДокументов.УдалитьКолонку(2);
```

УстановитьПараметрыКолонки

Устанавливает новые значения параметров колонки.

Синтаксис:

```
УстановитьПараметрыКолонки(<Колонка>, <Тип>, <Длина>, <Точность>, <Заголовок>,
                           <Ширина>, <Формат>, <Положение>)
```

Англоязычный синоним:

```
SetColumnParameters
```

Параметры:

<Колонка>	Номер или идентификатор колонки, для которой будут установлены новые параметры.
<Тип>	Необязательный параметр. Строка, описывающая тип колонки или вид субконто.
<Длина>	Необязательный параметр. Длина для строковых и числовых значений.
<Точность>	Необязательный параметр. Точность для числовых значений.
<Заголовок>	Необязательный параметр. Заголовок колонки для показа.
<Ширина>	Необязательный параметр. Ширина колонки в таблице.
<Формат>	Необязательный параметр. Строковое выражение, содержащее форматную строку, которая будет использована при визуальном отображении значений данной колонки.
<Положение>	Необязательный параметр. Определяет вариант выравнивания при визуальном отображении значений данной колонки. Число: 1 — слева; 2 — справа.

Описание:

Метод УстановитьПараметрыКолонки устанавливает новые значения параметров колонки (только те которые указаны). Если какой либо параметр при вызове метода не задан, то данный параметр колонки не изменяется.

Пример:

```
Табл.УстановитьПараметрыКолонки("Код", "Число", 6, 0, "Код", 6);
```

ПолучитьПараметрыКолонки

Возвращает значения параметров колонки.

Синтаксис:

```
ПолучитьПараметрыКолонки(<Колонка>, <Тип>, <Длина>, <Точность>, <Заголовок>,
                         <Ширина>, <Формат>, <Положение>)
```

Англоязычный синоним:

```
GetColumnParameters
```

Параметры:

<Колонка>	Номер или идентификатор колонки, для которой требуется получить параметры.
<Тип>	Необязательный параметр. Идентификатор переменной, в которую метод вернет строку, описывающую тип колонки.
<Длина>	Необязательный параметр. Идентификатор переменной, в которую метод вернет длину для строковых и числовых значений.
<Точность>	Необязательный параметр. Идентификатор переменной, в которую метод вернет точность для числовых значений колонки.
<Заголовок>	Необязательный параметр. Идентификатор переменной, в которую метод вернет строку, описывающую заголовок колонки для показа.
<Ширина>	Необязательный параметр. Идентификатор переменной, в которую метод вернет ширину колонки в таблице.
<Формат>	Необязательный параметр. Строковое выражение, содержащее форматную строку, которая будет использована при визуальном отображении значений данной колонки.

<Положение> Необязательный параметр. Определяет вариант выравнивания при визуальном отображении значений данной колонки. Число: 1 — слева; 2 — справа.

Возвращаемое значение:

Номер или идентификатор колонки. Если в параметре <Колонка> задан номер колонки, то возвращается идентификатор колонки, и наоборот.

Описание:

Метод ПолучитьПараметрыКолонки возвращает значения параметров колонки по номеру колонки или по ее идентификатору.

Пример:

```
Перем ВыбТип;
Перем ВыбДлина;
Перем ВыбТочность;
Перем ВыбЗаголовок;
Перем ВыбШирина;
Табл.ПолучитьПараметрыКолонки("Код", ВыбТип, ВыбДлина, ВыбТочность, ВыбЗаголовок,
                                ВыбШирина);
```

КоличествоСтрок

Устанавливает/возвращает количество строк в таблице значений,

Синтаксис:

КоличествоСтрок(<НовоеКоличествоСтрок>)

Англоязычный синоним:

LinesCnt

Параметры:

<НовоеКоличествоСтрок> Необязательный параметр. Новое количество строк в таблице значений.

Возвращаемое значение:

Числовое значение — количество строк в таблице значений до вызова метода.

Описание:

Метод КоличествоСтрок устанавливает/возвращает количество строк в таблице значений

Пример:

```
Разм = ТаблицаТоваров.КоличествоСтрок();
```

НоваяСтрока

Добавляет новую строку в таблицу значений.

Синтаксис:

НоваяСтрока(<НомерСтроки>)

Англоязычный синоним:

NewLine

Параметры:

<НомерСтроки> Необязательный параметр. Числовое выражение, содержащее позицию, в которую следует вставить новую строку.

Возвращаемое значение: номер новой строки.

Описание:

Метод НоваяСтрока вставляет новую строку в таблицу значений в указанную позицию.

Пример:

```
Табл.НоваяСтрока(1);
```

УдалитьСтроку

Удаляет строку из таблицы значений.

Синтаксис:

УдалитьСтроку(<НомерСтроки>)

Англоязычный синоним:

DeleteLine

Параметры:

<НомерСтроки> Необязательный параметр. Номер строки. Если не указан, то удаляется текущая строка

Описание:

Метод УдалитьСтроку удаляет строку из таблицы значений.

Пример:

```
ТаблДокументов.УдалитьСтроку(2);
```

УдалитьСтроки

Удаляет все строки из таблицы значений.

Синтаксис:

УдалитьСтроки()

Англоязычный синоним:

DeleteLines

Описание:

Метод УдалитьСтроки удаляет все строки из таблицы значений.

Пример:

```
ТаблДокументов.УдалитьСтроки();
```

ВыбратьСтроки

Открывает выборку строк таблицы значений.

Синтаксис:

```
ВыбратьСтроки()
```

Англоязычный синоним:

```
SelectLines
```

Описание:

Метод ВыбратьСтроки предоставляет возможность перебирать строки таблицы значений (открывает выборку). Дальнейшая выборка осуществляется при помощи метода ПолучитьСтроку.

Пример:

```
Табл.ВыбратьСтроки();
Пока Табл.ПолучитьСтроку() = 1 Цикл
    ...
КонецЦикла;
```

ПолучитьСтроку

Получить из выборки следующую строку таблицы значений.

Синтаксис:

```
ПолучитьСтроку()
```

Англоязычный синоним:

```
GetLine
```

Возвращаемое значение:

Число: 1 — если строка выбрана успешно, 0 — если строка не выбрана (выборка закончилась).

Описание:

Метод ПолучитьСтроку выбирает следующую строку таблицы значений в последовательности выборки, открытой перед этим при помощи метода ВыбратьСтроки.

Пример:

```
Табл.ВыбратьСтроки();
Пока Табл.ПолучитьСтроку() = 1 Цикл
    ...
КонецЦикла;
```

ВыбратьСтроку

Открыть окно для интерактивного выбора строки в таблице значений.

Синтаксис:

```
ВыбратьСтроку(<Строка>, <Заголовок>, < Таймаут>)
```

Англоязычный синоним:

```
ChooseLine
```

Параметры:

<Строка>	Необязательный параметр. Идентификатор переменной, куда помещается значение — номер выбранной строки. При вызове метода здесь можно передавать значение начального номера строки.
<Заголовок>	Необязательный параметр. Строковое выражение, значение которого отображается в заголовке диалогового окна. Может использоваться для подсказки пользователю.
<Таймаут>	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя. Если не задано, то время ожидания бесконечно.

Возвращаемое значение:

Числовое значение: 1 — если выбор произведен (нажата кнопка «ОК»); 0 — если выбор не произведен (нажата кнопка «ОТМЕНА»); -1 (минус единица) — закончилось время <Таймаут> ожидания отклика пользователя.

Описание:

Метод ВыбратьСтроку вызывает диалоговое окно для интерактивного выбора строки из заранее подготовленной таблицы значений. Параметр <Строка> указывает на ту строку таблицы значений, на которой будет установлен курсор при открытии окна выбора. В этот же параметр возвращается выбранное значение. Если выбор отменен, то значение параметров не меняются.

Пример:

```
Перем НомСтроки;
Табл.ВыбратьСтроку(НомСтроки, "Выберите строку");
```

ПолучитьСтрокуПоНомеру

Получить строку таблицы значений по номеру.

Синтаксис:

```
ПолучитьСтрокуПоНомеру(<НомерСтроки>)
```

Англоязычный синоним:

GetLineByNumber

Параметры:

<НомерСтроки>	Номер строки, на которую следует переместиться.

Описание:

Метод ПолучитьСтрокуПоНомеру получает строку таблицы значений по заданному номеру. Указанная строка становится текущей.

Пример:

```
Табл.ПолучитьСтрокуПоНомеру(3);
```

СдвинутьСтроку

Переместить строку таблицы значений на новую позицию.

Синтаксис:

```
СдвинутьСтроку(<КоличествоСтрок>, <НомерСтроки>)
```

Англоязычный синоним:

```
MoveLine
```

Параметры:

<КоличествоСтрок>	Число строк, на которое надо переместить строку. Если число положительное, то строка сдвигается вниз, если отрицательное, то вверх.
<НомерСтроки>	Необязательный параметр. Номер строки, которую надо переместить, если не задан, то текущая.

Возвращаемое значение: новая позиция строки

Описание:

Метод СдвинутьСтроку перемещает строку таблицы значений на новую позицию.

Пример:

```
Табл.СдвинутьСтроку(3, 5);
```

УстановитьЗначение

Установить значение конкретной колонки в данной строке таблицы значений.

Синтаксис:

```
УстановитьЗначение(<Строка>, <Колонка>, <Знач>)
```

Англоязычный синоним:

```
SetValue
```

Параметры:

<Строка>	Номер строки.
<Колонка>	Номер или идентификатор колонки.
<Знач>	Устанавливаемое значение.

Описание:

Метод УстановитьЗначение устанавливает значение конкретной колонки в данной строке таблицы значений.

Пример:

```
Табл.УстановитьЗначение(3, 5, ВыбЗнач);
```

ПолучитьЗначение

Получить значение конкретной колонки в данной строке таблицы значений.

Синтаксис:

```
ПолучитьЗначение(<Строка>, <Колонка>)
```

Англоязычный синоним:

```
GetValue
```

Параметры:

<Строка>	Номер строки.
<Колонка>	Номер или идентификатор колонки.

Возвращаемое значение:

Значение конкретной колонки в данной строке.

Описание:

Метод ПолучитьЗначение получает значение конкретной колонки в данной строке таблицы значений.

Пример:

```
ВыбЗнач = Табл.ПолучитьЗначение(3, 5);
```

НайтиЗначение

Найти заданное значение в таблице значений.

Синтаксис:

```
НайтиЗначение(<Знач>, <Строка>, <Колонка>)
```

Англоязычный синоним:

```
FindValue
```

Параметры:

<Знач>	Значение для поиска.
<Строка>	Идентификатор переменной, куда возвращается номер найденной строки. Если при вызове метода

передать в этот параметр номер строки, то поиск будет осуществляться только по указанной строке.

<Колонка> Идентификатор переменной, куда возвращается номер найденной колонки. Если при вызове метода передать в этот параметр номер или идентификатор колонки, то поиск будет осуществляться только по указанной колонке.

Возвращаемое значение:

Число: 0 — значение не найдено; 1 — значение найдено.

Описание:

Метод НайтиЗначение позволяет найти искомое значение в таблице значений и определить номер строки и номер колонки таблицы значений.

Пример:

```
Перем НомСтр;
Перем НомКолонки;
Табл.НайтиЗначение(ВыбЗнач, НомСтр, НомКолонки);
```

Сортировать

Сортировать таблицу значений по колонкам.

Синтаксис:

```
Сортировать(<Колонки>, <ДокументыПоДате>)
```

Англоязычный синоним:

```
Sort
```

Параметры:

<Колонки> Строковое выражение, которое определяет колонки, порядок и направление сортировки. Формат передаваемой строки — это разделенные запятыми номера или идентификаторы колонок со знаком направления сортировки ("+" по возрастанию, "-" по убыванию, "*" по внутреннему значению). Знак направления сортировки следует указывать до обозначения колонки через пробел или без пробела. По умолчанию направление сортировки принимается по возрастанию. Например: "-Код, +Цена, -8, 5".

<ДокументыПоДате> Необязательный параметр. Имеет смысл только в том случае, если значениями таблицы значений являются документы. В этом случае можно задавать сортировку документов по их хронологии. Число: 1 — сортировка по хронологии документов; 0 — нет. Значение по умолчанию — 0.

Описание:

Метод Сортировать выполняет сортировку таблицы значений по заданным колонкам, порядку и направлению сортировки.

Пример:

```
Табл.Сортировать("-Код, +Цена, -8, 5");
```

Очистить

Очистить таблицу значений и удалить колонки.

Синтаксис:

```
Очистить()
```

Англоязычный синоним:

```
Clear
```

Описание:

Метод Очистить выполняет очистку таблицы значений и удаляет колонки.

Пример:

```
Табл.Очистить();
```

Итог

Вычислить сумму по колонке таблицы значений.

Синтаксис:

```
Итог(<Колонка>)
```

Англоязычный синоним:

```
Total
```

Параметры:

<Колонка> Номер или идентификатор колонки, по которой считать сумму.

Возвращаемое значение:

Сумма по колонке.

Описание:

Метод Итог вычисляет сумму по заданной колонке таблицы значений.

Пример:

```
Табл.Итог(2);
```

Заполнить

Заполнить значения конкретных колонок в строках таблицы значений переданным значением.

Синтаксис:

```
Заполнить(<Знач>, <НачСтрока>, <КонСтрока>, <Колонки>)
```

Англоязычный синоним:

```
Fill
```

Параметры:

<Знач>	Одиночное значение или список значений или таблица значений.
<НачСтрока>	Необязательный параметр. Номер начальной строки, с которой надо начинать заполнение. Значение по умолчанию 1.
<КонСтрока>	Необязательный параметр. Номер последней строки, по которую надо заполнять, если не указана то до последней.
<Колонки>	Необязательный параметр. Номера или идентификаторы колонок, которые надо заполнять. Если параметр не задан, то заполняются все колонки.

Описание:

Метод Заполнить позволяет заполнить значения конкретных колонок в строках таблицы значений переданным значением.

Пример:

```
Табл.Заполнить(ВыбЗнач, 2, 5, "5, 6, 7");
```

Свернуть

Свернуть таблицу значений но соответствующим значениям колонок.

Синтаксис:

```
Свернуть(<ГруппКолонки>, <СуммКолонки>)
```

Англоязычный синоним:

```
GroupBy
```

Параметры:

<ГруппКолонки>	Группировочные колонки (номера или идентификаторы колонок через запятую), по которым группировать данные.
<СуммКолонки>	Суммируемые колонки (номера или идентификаторы колонок через запятую), по которым суммировать данные.

Описание:

Метод Свернуть позволяет свернуть таблицу значений по соответствующим значениям колонок, т. е. заменяет на одну строку все дублирующие (по значениям группировочных колонок) строки, суммируя значения но суммируемым колонкам.

Пример:

```
Табл.Свернуть("1,2,3,4", "5,6,7");
```

Выгрузить

Выгрузить данные из таблицы значений.

Синтаксис:

```
Выгрузить(<Знач>, <НачСтрока>, <КонСтрока>, <Колонки>)
```

Англоязычный синоним:

```
Unload
```

Параметры:

<Знач>	Идентификатор переменной, содержащей значение типа «Таблица значений» или «Список значений», в которое нужно выгрузить данные. Если переданное значение пустое, тогда система сама создаст объект типа «Таблица значений».
<НачСтрока>	Необязательный параметр. Номер начальной строки, с которой надо начинать выгрузку. Значение по умолчанию 1.
<КонСтрока>	Необязательный параметр. Номер последней строки, по которую надо выгружать, если не указана, то до последней.
<Колонки>	Необязательный параметр. Номера или идентификаторы колонок, которые надо выгружать. Если параметр не задан, то выгружаются все колонки.

Описание:

Метод Выгрузить позволяет выгрузить заданную область таблицы значений в переданное значение. Если в качестве значения для выгрузки задан список значений, то система выгружает данные из таблицы значений по колонкам.

Пример:

```
НовТабл = СоздатьОбъект("ТаблицаЗначений");
СтарТабл.Выгрузить(НовТабл, 1, 5, "1,3,5,7");
```

Загрузить

Скопировать структуру и значения таблицы значений.

Синтаксис:

```
Загрузить(<ТаблицаЗначений>)
```

Англоязычный синоним:

```
Load
```

Параметры:

<ТаблицаЗначений>	Значение типа «Таблица значений». Структура и значения для загрузки.

Описание:

Метод Загрузить позволяет скопировать структуру и значения таблицы значений. Прежняя структура колонок таблицы значений при этом очищается.

Пример:

```
НовТабл = СоздатьОбъект("ТаблицаЗначений");
НовТабл.Загрузить(СтарТабл);
```

ВидимостьКолонки

Показать/скрыть колонки таблицы значений.

Синтаксис:

ВидимостьКолонки(<Колонки>, <Видимость>, <Позиция>)

Англоязычный синоним:

ColumnVisibility

Параметры:

<Колонки>	Строковое выражение, которое определяет список колонок. Формат передаваемой строки — это разделенные запятыми номера или идентификаторы колонок, для которых применяется данный метод. Например: «Код, Цена, 8, 5».
<Видимость>	Необязательный параметр. Число: 1 — показать колонки; 0 — скрыть. По умолчанию — 1.
<Позиция>	Необязательный параметр. Позиция, в которой показывать колонку. Если параметр не задан, то колонки отображаются в соответствии с порядком колонок в таблице.

Возвращаемое значение:

Если передана одна колонка, то возвращается значение видимости колонки до вызова метода. Число: 1 — колонка показана; 0 — колонка скрыта.

Описание:

Метод ВидимостьКолонки определяет перечень колонок и их видимость в визуальном представлении таблицы значений.

Пример:

```
ТаблДиалога.ВидимостьКолонки("Код, Цена, 8, 5");
```

ТекущаяСтрока

Установить/определить текущую строку таблицы в элементе диалога типа «ТаблицаЗначений».

Синтаксис:

ТекущаяСтрока(<ИндексСтроки>)

Англоязычный синоним:

CurrentLine

Параметры:

<ИндексСтроки>	Необязательный параметр. Числовое выражение с задаваемым индексом строки для элемента диалога типа «ТаблицаЗначений», на которую требуется установить курсор. Если параметр не задан, то положение курсора в поле диалога не меняется.

Возвращаемое значение:

Числовое значение, соответствующее индексу текущей строки поля диалога (до его изменения) или 0, если текущей строки нет.

Описание:

Метод ТекущаяСтрока в тексте программного модуля можно использовать синтаксически как функцию или процедуру. Данный метод позволяет установить и/или считать текущее положение курсора в элементе диалога типа «ТаблицаЗначений».

Данный метод можно использовать только для объектов, которые созданы при помощи визуальных средств конфигуратора (в форму вставлены элементы диалога «ТаблицаЗначений», а идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа «ТаблицаЗначений»).

Пример:

```
ТаблДиалога.ТекущаяСтрока(2);
```

ТекущаяКолонка

Установить/определить текущую колонку таблицы в элементе диалога типа «ТаблицаЗначений».

Синтаксис:

ТекущаяКолонка(<НоваяКолонка>, <ТекущаяКолонка>)

Англоязычный синоним:

CurrentColumn

Параметры:

<НоваяКолонка>	Необязательный параметр. Номер или идентификатор колонки для элемента диалога типа «ТаблицаЗначений», на которую требуется установить курсор. Если параметр не задан, то текущая колонка в поле диалога не меняется.
<ТекущаяКолонка>	Необязательный параметр. Идентификатор переменной, куда система возвращает номер текущей колонки.

Возвращаемое значение:

Идентификатор текущей колонки поля диалога (до его изменения) или "" (пустая строка), если ее нет.

Описание:

Метод ТекущаяКолонка в тексте программного модуля можно использовать синтаксически как функцию или процедуру. Данный метод позволяет установить и/или считать текущее положение курсора в элементе диалога типа «ТаблицаЗначений».

Данный метод можно использовать только для объектов, которые созданы при помощи визуальных средств конфигуратора (в форму вставлены элементы диалога «ТаблицаЗначений», а идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа «ТаблицаЗначений»).

Пример:

```
Перем НомКолонки;
ТаблДиалога.ТекущаяКолонка(2, НомКолонки);
```

Фиксировать

Фиксировать в элементе диалога типа «ТаблицаЗначений» колонки и строки.

Синтаксис:

```
Фиксировать(<КолСтрок>, <КолКолонок>)
```

Англоязычный синоним:

```
Fix
```

Параметры:

<КолСтрок>	Необязательный параметр. Количество фиксируемых строк. Если не указаны, то не изменять фиксацию.
<КолКолонок>	Необязательный параметр. Количество фиксируемых колонок. Если не указаны, то не изменять фиксацию.

Описание:

Метод Фиксировать позволяет фиксировать в элементе диалога типа «ТаблицаЗначений» колонки и строки.

Пример:

```
ТаблДиалога.Фиксировать(1, 1);
```

ВыводитьПиктограммы

Выводить в элементе диалога типа «ТаблицаЗначений» пиктограммы.

Синтаксис:

```
ВыводитьПиктограммы(<Колонка>, <Пиктограмма>)
```

Англоязычный синоним:

```
ShowImages
```

Параметры:

<Колонка>	Номер или идентификатор колонки, которая содержит номера пиктограмм.
<Пиктограмма>	Необязательный параметр. Начальный номер пиктограммы. Значение по умолчанию: 1.

Описание:

Метод ВыводитьПиктограммы устанавливает режим, при котором в колонке выводится не текст, а пиктограмма. Пиктограмма будет браться из картинки, назначенной элементу диалога «ТаблицаЗначений» в закладке «Картинка» в конфигураторе. Картинка должна быть формата .bmp, содержать все пиктограммы для этой таблицы значений, и состоять из последовательности пиктограмм размером 16x15 пикселей. Пиктограммы будут выбираться из картинки по номеру, взятому из числового значения данной колонки в текущей строке. Параметр <Пиктограмма> позволяет установить для данной колонки начальный номер пиктограмм в картинке.

Пример:

```
ТаблДиалога.ВыводитьПиктограммы(1, 1);
```

Пример использования объекта Таблица Значений

Пример:

```
Функция ВыбратьОплаты(ВыбДок)
   ТЗ = СоздатьОбъект("ТаблицаЗначений");
   ТЗ.НоваяКолонка("ДатаДок", "Дата", , , "Дата", 15);
   ТЗ.НоваяКолонка("Номер", "Строка", 20, , "Номер", 15);
   ТЗ.НоваяКолонка("Сумма", "Число", 18, 2, "Сумма", 20);
   ТЗ.НоваяКолонка("РасчетныйСчет", "Справочник.БанковскиеСчета", , ,
                   "Расч.счет", 30);
   ТЗ.НоваяКолонка("ПодчДок", "Документ", , , );
   ТЗ.ВидимостьКолонки("ПодчДок", 0);
   Док = СоздатьОбъект("Документ");
   Док.ВыбратьПодчиненныеДокументы(, , ВыбДок);
   Пока Док.ПолучитьДокумент() = 1 Цикл
      Если Док.Вид() = "ОплатаПоставщикам" Тогда
         ТЗ.НоваяСтрока();
         ТЗ.ДатаДок = Док.ДатаДок;
         ТЗ.Номер = Док.НомерДок;
         ТЗ.Сумма = Док.Итог("Сумма");
         ТЗ.РасчетныйСчет = Док.РасчетныйСчет;
         ТЗ.ПодчДок = Док.ТекущийДокумент();
      КонецЕсли;
```

```
    КонецЦикла;
    ВыбСтрока = 0;
    Если ТЗ.ВыбратьСтроку(ВыбСтрока, "Выберите оплату") = 1 Тогда
       Возврат ТЗ.ПолучитьЗначение(ВыбСтрока, "ПодчДок");
КонецФункции
```

Глава 30
Атрибуты и методы контекста Модуля формы

Описанные в данной главе атрибуты и методы доступны только в контексте Модуля формы (см. «Виды программных модулей»). К Модулям форм относятся практически все программные модули (исключение составляют Глобальный модуль, Модуль документа, Модуль вида расчета).

Атрибуты контекста Модуля формы

СтрокаДействийФормы

Содержит командную строку системных действий в кнопке формы.

Синтаксис:

`СтрокаДействийФормы`

Англоязычный синоним:

`StringFormActions`

Описание:

Атрибут `СтрокаДействийФормы` предоставляет доступ к значению командной строки кнопки формы (строка системных команд в свойствах кнопки после символа #). Данный атрибут доступен в процедуре, вызываемой перед знаком # из строки формулы кнопки (кнопка — «Свойства» — «Дополнительные» — «Формула»).

Доступ к данному атрибуту возможен только в контексте Модуля формы.

Пример:

* Допустим, что в некоторой форме в свойствах какой-либо кнопки записана следующая формула:

```
УстКоманд() # Записать? Провести? Закрыть?
```

В программном модуле можно записать процедуру, которая будет устанавливать значение командной строки действий:

```
Процедура УстКоманд()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      СтрокаДействийФормы = "Записать? Закрыть?";
   ИначеЕсли НазваниеНабораПрав() = "Менеджер" Тогда
      СтрокаДействийФормы = "Записать? Провести? Закрыть?";
   Иначе
      СтрокаДействийФормы = "Закрыть?";
КонецПроцедуры
```

Форма

Объект агрегатного типа «Форма».

Синтаксис:

`Форма`

Англоязычный синоним:

`Form`

Описание:

Атрибут `Форма` представляет собой ссылку на объект типа «Форма». Атрибуты и методы объекта «Форма» позволяют в программном модуле изменять свойства визуального отображения диалога в целом (такие как `Заголовок`, `Закладки` и т. п.), а также свойства визуального отображения отдельных элементов диалога.

В тексте программного модуля через точку после имени атрибута «Форма» можно записывать идентификаторы элементов диалога, а далее через точку можно вызывать методы управления свойствами этих элементов.

Доступ к данному атрибуту возможен только в контексте Модуля формы.

Пример:

```
Форма.ТолькоПросмотр(0);
Форма.Prompt.Видимость(0);
```

Атрибуты объекта Форма

Закладки

Объект типа «СписокЗначений», содержащий описания закладок текущей формы.

Синтаксис:

`Закладки`

Англоязычный синоним:

`TabCtrl`

Описание:

Атрибут объекта «Форма» Закладки представляет собой объект типа «СписокЗначений», содержащий описания закладок текущей формы. Данный атрибут предоставляет возможность программно манипулировать порядком и названиями закладок формы. В тексте программного модуля через точку после имени объекта «Форма» можно вызвать атрибут «Закладки», а далее через точку можно вызывать методы управления закладками как объектом типа «СписокЗначений».

Доступ к данному атрибуту возможен только в контексте Модуля формы после применения метода ИспользоватьЗакладки.

Пример:

```
Процедура ОбработкаКлиектов()
   Если ФлагФранчайзи = 1 Тогда
      Если Форма.Закладки.НайтиЗначение(2) = 0 Тогда
         Форма.Закладки.ДобавитьЗначение(2, "Поставщики");
      КонецЕсли;
   Иначе
      Позиция = Форма.Закладки.НайтиЗначение(2);
      Если Позиция > 0 Тогда
         Форма.Закладки.УдалитьЗначение(Позиция);
      КонецЕсли;
   КонецЕсли;
   форма.Закладки.Сортировать();
КонецПроцедуры
```

См. также: ИспользоватьЗакладки, ПриВыбореЗакладки, «Работа со СпискомЗначений»

<ЭлементДиалога>

Объект агрегатного типа «*элемент диалога*».

Синтаксис:

```
<ЭлементДиалога>
```

Описание:

Значение атрибута объекта «Форма» <ЭлементДиалога> представляет собой ссылку на элемент диалога. Элементы диалога — это объекты специального агрегатного типа. Методы элементов диалога позволяют в программном модуле изменять их свойства, такие как Цвет, Видимость, Доступность.

В тексте программного модуля через точку после имени объекта «Форма» можно вызвать атрибут <ЭлементДиалога>, а далее через точку можно вызывать методы управления их свойствами. В тексте программного модуля в качестве названия данного атрибута подставляется идентификатор конкретного элемента диалога, существующего в диалоговом окне формы.

Пример:

```
// в форме диалога существуют элементы
// НазвФирмы, ИнформПоле, Название
// Поменяем свойства этих элементов диалога
Форма.НазвФирмы.Доступность(0);
Форма.ИнформПоле.Видимость(0);
Форма.Название.Цвет(255, 0, 0);
```

См. также: Цвет, Видимость, Доступность

Параметр

Параметр, переданный при программном открытии формы.

Синтаксис:

```
Параметр
```

Англоязычный синоним:

```
Parameter
```

Описание:

Атрибут Параметр объекта «Форма» представляет собой значение, переданное данной форме при ее открытии программно с помощью методов Открытьформу, ОткрытьФормуМодально, ОткрытьПодбор. Если форма открыта интерактивно или если параметр при программном открытии не передавался, то данный атрибут содержит значение неопределенного типа.

Доступ к данному атрибуту возможен только в контексте Модуля формы.

Пример:

```
ПереданныйПараметр = Форма.Параметр;
```

См. также: ОткрытьФорму, ОткрытьФормуМодально, ОткрытьПодбор

Методы объекта Форма

При помощи объекта «Форма» предоставляется возможность управлять визуальным представлением текущей формы, опрашивать текущее состояние режима работы формы и т. п.

ТолькоПросмотр

Установка режима редактирования формы,

Синтаксис:

```
ТолькоПросмотр(<Режим>)
```

Англоязычный синоним:

```
Readonly
```

Параметры:

<Режим> Необязательный параметр. Числовое выражение:
0 — разрешено редактирование элементов формы;
1 — запрещено редактирование.

Возвращаемое значение:

Текущее числовое значение режима редактирования формы (на момент до исполнения метода).

Описание:

Метод ТолькоПросмотр позволяет установить режим редактирования текущей формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.ТолькоПросмотр(0);
```

Обновить

Обновить форму.

Синтаксис:

```
Обновись(<Флаг>)
```

Англоязычный синоним:

```
Refresh
```

Параметры:

<Флаг> Числовое выражение:
1 — установить флаг модифицированности (признак изменения реквизитов текущей формы справочника или документа);
0 — не устанавливать флаг модифицированности.

Описание:

Метод Обновить выполняет обновление окна формы. Кроме того, этот метод позволяет программно установить флаг модифицированности формы. Данный метод имеет смысл вызывать, если через контекст формы (извне, из другого программного модуля) были произведены изменения реквизитов формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.Обновить(1)
```

См. также: ОткрытьФорму, ОткрытьПодбор, Модифицированность

ИспользоватьЗакладки

Установить режим использования закладок в форме.

Синтаксис:

```
ИспользоватьЗакладки(<Флаг>)
```

Англоязычный синоним:

```
TabCtrlState
```

Параметры:

<Флаг> Необязательный параметр. Числовое выражение:
1 — включить закладки в форме;
0 — выключить закладки в форме.

Возвращаемое значение:

Текущее значение режима использования закладок формы (на момент до исполнения метода).

Описание:

Метод ИспользоватьЗакладки устанавливает режим использования закладок в форме.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.ИспользоватьЗакладки(1)
```

См. также: Закладки, ПриВыбореЗакладки, ИспользоватьСлой

ИспользоватьСлой

Установить режим отображения слоя в форме.

Синтаксис:

```
ИспользоватьСлой(<ИмяСлоя>, <Режим>)
```

Англоязычный синоним:

```
UseLayer
```

Параметры:

<ИмяСлоя> Строковое выражение — название слоя формы как оно задано в конфигураторе. Параметр может быть составным (указывать несколько слоев). В этом случае имена слоев перечисляются через запятую.

<Режим> Необязательный параметр. Числовое выражение:
0 — скрыть слой <ИмяСлоя> в форме;
1 — показать слой <ИмяСлоя> в форме;
2 — показать слой <ИмяСлоя> и скрыть все остальные.
Значение по умолчанию — 2.

Описание:

Метод Использовать Слой устанавливает режим отображения слоя в форме. Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.ИспользоватьСлой("Основной, КнопкиДиалога, Дополнительный", 1);
```

См. также: Закладки, ПриВыбореЗакладки

Заголовок

Установка заголовка окна формы.

Синтаксис:

```
Заголовок(<Название>, <Режим>)
```

Англоязычный синоним:

```
Caption
```

Параметры:

<Название>	Строковое выражение — новый заголовок окна формы.
<Режим>	Числовое выражение: 0 — выводит заданный заголовок вместо стандартного заголовка окна; 1 — выводит заданный загловок вместе со стандартным заголовком окна.

Возвращаемое значение:

Текущий заголовок окна формы, установленный этой функцией.

Описание:

Метод Заголовок позволяет установить новый заголовок окна формы. Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.Заголовок("Накладная №" + Док.НомерДок, 0 );
```

ПанельИнструментов

Управление видимостью панели инструментов окна формы.

Синтаксис:

```
ПаяельИнструментов(<Режим>)
```

Англоязычный синоним:

```
ToolBar
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — показать панель инструментов; 0 — не показывать панель инструментов. Если параметр не задан, то метод просто возвращает текущее состояние видимости панели инструментов.

Возвращаемое значение:

Текущее состояние видимости панели инструментов окна формы.

Описание:

Метод ПанельИнструментов предоставляет возможность управления видимостью панели инструментов окна формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.ПанельИнструментов(0);
```

КнопкаПоУмолчанию

Задает кнопку в форме, которая будет «срабатывать» при нажатии комбинации клавиш Ctrl + Enter.

Синтаксис:

```
КнопкаПоУмолчанию(<ИдентификаторКнопки>)
```

Англоязычный синоним:

```
DefButton
```

Параметры:

<ИдентификаторКнопки>	Строковое выражение — идентификатор одной из кнопок формы объекта метаданных.

Описание:

Метод КнопкаПоУмолчанию позволяет назначить одну из кнопок формы кнопкой «по умолчанию»: в этом случае нажатие комбинации клавиш Ctrl+Enter приведет к такому же результату, как и нажатие такой кнопки мышью.

В качестве параметра методу передается идентификатор одной из кнопок формы — так, как он указан в палитре свойств соответствующего элемента диалога типа «Кнопка».

Пример:

* В модуле формы документа «ПлатежноеПоручение» кнопкой по умолчанию назначается кнопка с идентификатором " ОК".

```
Процедура ПриОткрытии()
   Форма.КнопкаПоУмолчанию("ОК");
КонецПроцедуры
```

ОбработкаВыбораСтроки

Включает в форме списка обработку выбора строки предопределенной процедурой ПриВыбореСтроки.

Синтаксис:
ОбработкаВыбораСтроки(<Флаг>)
Англоязычный синоним:
ProcessSelectLine
Параметры:

<Флаг>	Числовое выражение: 1 — включает обработку выбора предопределенной процедурой; 0 — выключает обработку выбора предопределенной процедурой.

Описание:
Метод ОбработкаВыбораСтроки включает в форме списка (справочника, журнала, счетов, журнала операций, журнала проводок) обработку выбора строки предопределенной процедурой ПриВыбореСтроки.
Доступ к данному методу возможен только в контексте Модуля формы списка.
Пример:
```
Форма.ОбработкаВыбораСтроки(1)
```
См. также: ПриВыбореСтроки

ВыполнитьВыбор

Осуществляет выбор из формы подбора.
Синтаксис:
ВыполнитьВыбор(<Знач>)
Англоязычный синоним:
MakeChoise
Параметры:

<Знач>	Значение, которое выбирается в форме, открытой для подбора или выбора значения.

Описание:
Метод ВыполнитьВыбор осуществляет выбор в форме, открытой для подбора или выбора значения (аналогично интерактивному двойному щелчку в подборе). Метод предназначен, прежде всего, для организации подбора из формы отчета.
Доступ к данному методу возможен только в контексте Модуля формы.
Пример:
```
// осуществим выбор в форме журнала документов
Форма.ВыполнитьВыбор(ТекущийДокумент);
```

РежимВыбора

Возвращает режим работы формы.
Синтаксис:
РежимВыбора()
Англоязычный синоним:
ChoiseMode
Возвращаемое значение:
Число: 0 — форма открыта не для выбора; 1 — форма открыта для выбора одного значения; 2 — форма открыта для выбора нескольких значений.
Описание:
Метод РежимВыбора возвращает режим работы формы. Доступ к данному методу возможен только в контексте Модуля формы.
Пример:
```
РежимРаботыФормы = Форма.РежимВыбора();
```

МодальныйРежим

Возвращает режим работы формы.
Синтаксис:
МодальныйРежим()
Англоязычный синоним:
ModalMode
Возвращаемое значение:
Число: 0 — немодальный режим; 1 — модальный режим.
Описание:
Метод МодальныйРежим возвращает режим работы формы. Доступ к данному методу возможен только в контексте Модуля формы.
Пример:
```
РежимРаботыФормы = Форма.МодальныйРежим();
```

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 = Форма.ПолучитьАтрибут("ЦенаРозн");
```

АктивныйЭлемент

Возвращает идентификатор активного элемента диалога.

Синтаксис:

```
АктивныйЭлемент()
```

Англоязычный синоним:

```
ActiveControl
```

Возвращаемое значение:

Строковое значение — идентификатор активного элемента диалога.

Описание:

Метод АктивныйЭлемент возвращает идентификатор активного элемента диалога.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
ИдАктЭлем = Форма.АктивныйЭлемент();
```

ТекущаяКолонка

Возвращает идентификатор текущей колонки многострочной части.

Синтаксис:

```
ТекущаяКолонка()
```

Англоязычный синоним:

```
CurrentColumn
```

Возвращаемое значение:

Строковое значение — идентификатор текущей колонки многострочной части.

Описание:

Метод ТекущаяКолонка возвращает идентификатор текущей колонки многострочной части.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
ИдТекКолонки = Форма.ТекущаяКолонка();
```

Закрыть

Закрыть форму.

Синтаксис:

```
Закрыть(<ЗапрСохр>)
```

Англоязычный синоним:

```
Close
```

Параметры:

<ЗапрСохр> Необязательный параметр. Число: 0 — закрыть форму без вопросов; 1 — если документ, счет, элемент справочника изменен, то будет запрос о сохранении изменений. Значение по умолчанию — 1.

Описание:

Метод Закрыть закрывает форму. Доступ к данному методу возможен только в контексте Модуля формы.

Замечание. Действие данного метода не прерывает выполнения текущей процедуры программы, т. е. процедура доработает до конца. Данный метод лишь устанавливает признак, который отрабатывает после окончания выполнения процедуры.

Пример:

```
Форма.Закрыть();
```

Методы элементов диалога

При помощи атрибута «*Форма*» средства языка предоставляют возможность программно управлять свойствами элементов диалога. В тексте программного модуля формы через точку после имени атрибута «*Форма*» можно записывать идентификаторы элементов диалога, а далее через точку можно вызывать методы управления свойствами этих элементов.

Доступ к данным методам возможен только в контексте Модуля формы.

Видимость

Установка режима отображения.

Синтаксис:

Видимость(<Режим>)

Англоязычный синоним:

Visible

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — элемент диалога отображается; 0 — элемент диалога скрыт (невидим).

Возвращаемое значение:

Текущее числовое значение режима видимости элемента формы или колонки многострочной части (на момент до исполнения метода).

Описание:

Метод Видимость позволяет установить режим отображения выбранного элемента формы или колонки многострочной части формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.ИнформПоле.Видимость(0);
```

Доступность

Установка режима редактирования.

Синтаксис:

Доступность(<Режим>)

Англоязычный синоним:

Enable

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — разрешено редактирование элемента формы; 0 — запрещено редактирование.

Возвращаемое значение:

Текущее числовое значение режима редактирования элемента формы (на момент до исполнения метода).

Описание:

Метод Доступность позволяет установить режим редактирования выбранного элемента формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.НазвФирмы.Доступность(0);
```

Редактирование

Определяет возможность редактирования значения элемента диалога.

Синтаксис:

Редактирование(<Флаг>)

Англоязычный синоним:

EnableEdit

Параметры:

<Флаг>	Число: 1 — значение элемента редактируется как обычно; 0 — значение не редактируется но может выбираться кнопкой выбора. Отличие от метода Доступность в том, что Доступность отключает и кнопку выбора.

Описание:

Метод Редактирование определяет возможность редактирования значения непосредственно в элементе диалога для полей ввода типа Число, Строка, Дата, Счет.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.КолонкаЦены.Редактирование(1);
```

Цвет

Установка режима отображения цвета.

Синтаксис 1:

Цвет(<Цвет>)

Синтаксис 2:

Цвет(<R>, <G>, <B>)

Англоязычный синоним:

Color

Параметры:

<Цвет>	Необязательный параметр. Числовое выражение, значение которого задает RGB цвет. Допустимые значения от 0 до 16'777'215. Значение -1 (минус единица) задает цвет, заданный для всей формы по умолчанию.
<R>	Числовое выражение, значение которого задает красную компоненту цвета. Допустимые значения от 0 до 255.
<G>	Числовое выражение, значение которого задает зеленую компоненту цвета. Допустимые значения от 0 до 255.

<B> Числовое выражение, значение которого задает синюю компоненту цвета. Допустимые значения от 0 до 255.

Возвращаемое значение:

Текущее числовое значение RGB-цвета элемента формы (на момент до исполнения метода).

Описание:

Метод Цвет позволяет установить режим отображения цвета выбранного элемента формы.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Процедура ОбработкаНаименования()
   Если (Наименование = "") Тогда
      Форма.Название.Цвет(255, 0, 0);
   Иначе
      Форма.Название.Цвет(-1);
   КонецЕсли
КонецПроцедуры
```

Маска

Установить/определить маску интерактивного ввода для элементов диалога типа «поле ввода».

Синтаксис:

Маска(<СтрокаМаски>)

Англоязычный синоним:

Mask

Параметры:

<СтрокаМаски> Необязательный параметр. Строковое выражение — посимвольная маска интерактивного ввода для строковых реквизитов диалога (аналогично установке в свойствах реквизита диалога в конфигураторе).

Возвращаемое значение:

Строковое значение — текущая маска интерактивного ввода для строковых реквизитов (на момент до исполнения метода).

Описание:

Метод Маска позволяет установить посимвольную маску интерактивного ввода для элементов диалога типа «поле ввода» (аналогично установке в свойствах реквизита диалога в конфигураторе).

В параметре <СтрокаМаски> допустимы следующие символы:

- ! — введенный символ преобразуется в верхний регистр;
- 9 — произвольный символ цифры;
- # — произвольный символ цифры или - (знак минус) или + (знак плюс) или пробел;
- N — любые алфавитно-цифровые символы (буквы или цифры);
- X (латинского алфавита) — произвольный символ;
- @ — любые алфавитно-цифровые символы (буквы или цифры) в верхнем регистре.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
форма.Телефон.Маска("999-99-99");
```

ВыборГруппы

Установить режим выборки групп для элемента диалога типа «справочник».

Синтаксис:

ВыборГруппы(<Режим>)

Англоязычный синоним:

SelectGroup

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 1 — выбирать группы; 0 — не выбирать группы. Значение по умолчанию — 1.

Возвращаемое значение:

Текущее числовое значение режима выборки групп для элемента диалога (на момент до исполнения метода).

Описание:

Метод ВыборГруппы устанавливает режим выборки групп для выбранного элемента диалога типа «справочник».

По умолчанию, выборка элементов справочников для реквизитов 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.

Особенно необходим данный метод в случае, когда необходимо установить возможность выбора групп в графе табличной части документа.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

***** Если в табличной части документа существует какой-либо реквизит, например, «Статус» и необходимо, чтобы он мог принимать значения как элемента, так и группы, то в форме документа этому реквизиту следует установить:

```
Форма.Статус.ВыборГруппы(1);
```

ВыполнятьФормулуТолькоПриИзменении

Установка режима выполнения формулы.

Синтаксис:

```
ВыполнятьФормулуТолькоПриИзменении(<Режим>)
```

Англоязычный синоним:

```
ProcessFormulaOnlyWhenChanged
```

Параметры:

<Режим>	Числовое выражение: 1 — устанавливает режим, при котором в табличной части документа формула («Свойства» — «Дополнительные» — «Формула») вызывается только при изменении значения поля, а не при переходе между полями; 0 — устанавливает режим, при котором в табличной части документа формула вызывается при изменении значения поля и при переходе между полями.

Описание:

Метод ВыполнятьФормулуТолькоПриИзменении позволяет изменить режим выполнения формулы выбранного поля табличной части документа. Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.Цена.ВыполнятьФормулуТолькоПриИзменении(1);
```

Заголовок

Устанавливает/выдает заголовок элемента диалога.

Синтаксис:

```
Заголовок(<Название>)
```

Англоязычный синоним:

```
Caption
```

Параметры:

<Название>	Строковое выражение — новый заголовок колонки многострочной части формы, кнопки, рамки группы, текста, флажка, переключателя.

Возвращаемое значение:

Текущий заголовок элемента диалога.

Описание:

Метод Заголовок позволяет установить/прочитать заголовок колонки многострочной части формы, копки, рамки группы, текста, флажка, переключателя.

Доступ к данному методу возможен только в контексте Модуля формы.

Пример:

```
Форма.КолонкаЦены.Заголовок("Цена");
```

УстановитьТип

Установить тип для элемента диалога неопределенного вида.

Синтаксис:

```
УстановитьТип(<Выражение>)
```

Англоязычный синоним:

```
AssignType
```

Параметры:

<Выражение>	Выражение. Тип значения этого выражения будет присвоен элементу диалога.

Описание:

Метод УстановитьТип позволяет установить тип для элемента диалога, которому в конфигураторе назначен тип «Неопределенный».

Данный метод доступен в контексте Модуля формы (см. «Виды программных модулей»).

Пример:

```
Форма.ВыбЗнач.УстановитьТип(Товар);
```

См. также: НазначитьТип, ТипЗначения, ТипЗначенияСтр

НазначитьТип

Назначить тип для элемента диалога неопределенного вида.

Синтаксис:

```
НазначитьТип(<ИмяТипа>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
AssignType
```

Параметры:

<ИмяТипа>	Строковое выражение — название типа данных, которое назначается элементу диалога. 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 элемента диалога, которому в конфигураторе назначен тип «Неопределенный».

Данный метод доступен в контексте Модуля формы (см. «Виды программных модулей»).

Пример:

```
Форма.ВыбКолич.НазначитьТип("Число", 15, 2);
Форма.ВыбДата.НазначитьТип("Дата");
Форма.ВыбТовар.НазначитьТип("Справочник.Товары");
Форма.ВыбДокум.НазначитьТип("Документ");
```

См. также: УстановитьТип, ТипЗначения, ТипЗначенияСтр

НеИзменятьВид

Позволяет запретить пользователю при выборе значения справочника, документа, счета неопределенного вида изменять вид.

Синтаксис:

```
НеИзменятьВид(<Признак>)
```

Англоязычный синоним:

```
FixKind
```

Параметры:

<Признак> Числовое выражение — признак запрета изменения вида. Может принимать значения:
0 — разрешить изменение вида при выборе значения;
1 — запретить изменение вида при выборе значения.

Возвращаемое значение:

Число — текущее значение признака.

Описание:

Метод НеИзменятьВид применяется для реквизитов диалога формы, имеющих тип справочника, документа, счета неопределенного вида. Он позволяет запретить при выборе значения изменение вида значения.

Предположим, в диалоге существует реквизит типа справочник неопределенного вида. Вид этого элемента устанавливается при помощи встроенного языка процедурой НазначитьВид. Метод НеИзменятьВид(1) позволяет запретить пользователю при выборе значения выбрать вид справочника.

Пример:

* Например, в некотором документе, для его реквизита «Контрагент» типа справочник неопределенного вида, назначается вид «Организации» или «Сотрудники» и запрещается интерактивное изменение вида пользователем.

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

Методы контекста Модуля формы

Описанные в данном разделе методы доступны только в контексте Модуля формы (см. «Виды программных модулей»). К модулям форм относятся практически все программные модули (исключение составляют Глобальный модуль, Модуль документа, Модуль вида расчета).

ОткрытьПодбор

Открыть форму для подбора значений.

Синтаксис:

```
ОткрытьПодбор(<ИмяОбъекта>, <ИмяФормы>, <КонтекстФормы>, <ФлагМножВыбора>,
              <ТекЗнач>)
```

Англоязычный синоним:

```
OpenPermanentChoice
```

Параметры:

<ИмяОбъекта> Строковое выражение — имя объекта агрегатного типа, форму списка которого требуется открыть для подбора. Можно указывать справочник, журнал, документ (при указании документа открывается форма журнала для указанного документа). Имя объекта задается в следующем виде:

- "Справочник.XXXXX";
- "Документ.XXXXX";
- "Отчет.XXXXXX";
- "Обработка.XXXXXX";
- "Журнал.XXXXX" , где XXXXX — имя вида соответствующего объекта, как он задан в конфигураторе, например: "Справочник.Товары";
- "Журнал.Подчиненные";

- "ЖурналОпераций.XXXXX", где XXXXX — форма журнала пераций.
- "ПланСчетов.XXXXX", где XXXXX — идентификатор плана счетов, как он задан в конфигураторе. Если XXXXX не задан, то открывается подбор из любого (всех) плана счетов.

<ИмяФормы> Строковое выражение — имя формы подбора, как она задана в конфигураторе. Поскольку и справочники и журналы могут иметь несколько форм представления, то этим параметром можно конкретно указать, какая из форм представления объекта вызывается для подбора значений.

<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 формы подбора (см. «Передача контекста в качестве параметра»). С помощью значения этого контекста можно затем произвольно манипулировать формой подбора, пока она открыта. Пока форма открыта, тип значения данного параметра равен 100 (см. ТипЗначения), если закрыта — 0.

<ФлагМножВыбора> Необязательный параметр. Число: 1 — выбор нескольких значений; 0 — выбор одного значения, после чего окно закрывается. Значение по умолчанию: 1.

<ТекЗнач> Необязательный параметр. В случае выбора из списка, здесь можно передать значение, на которое следует изначально установить курсор при открытии формы подбора.

Описание:

Метод ОткрытьПодбор, который доступен только в контексте Модуля формы, выполняет открытие формы для подбора значений. Используется, например, для подбора значения реквизита документа.

При открытии подбора по журналу подчиненных документов, т. е. когда первый параметр <ИмяОбъекта> имеет значение "Журнал.Подчиненные", документ владелец, по которому следует построить журнал подчиненных документов, передается после вызова метода через полученный контекст формы.

Пример:

```
Процедура Подбор()
   ГруппаДляВыбора = ТекущийЭлемент();
   Если ГруппаДляВыбора.ЭтоГруппа() = 0 Тогда
      ГруппаДляВыбора = ГруппаДляВыбора.Родитель;
   КонецЕсли;
   ОткрытьПодбор("Справочник.Товары", "ДляПодбора", КонтПодб, 0);
   // установим в форме подбора реквизит "ГруппаВыбора"
   // используя полученный контекст формы подбора
   КонтПодб.ГруппаВыбора.ВыборГруппы(1);
   КонтПодб.ГруппаВыбора = ГруппаДляВыбора;
   КонтПодб.Обновить(0);
   УстановитьЗначениеВПодборе("ГруппаВыбора", ГруппаДляВыбора);
КонецПроцедуры
```

См. также: ОбработкаПодбора, УстановитьЗначениеВПодборе

УстановитьЗначениеВПодборе

Установить значение некоторого реквизита диалога в окне, открытом для подбора значения.

Синтаксис:

```
УстановитьЗначениеВПодборе(<ИмяРеквизита>, <Значение>)
```

Англоязычный синоним:

```
SetValueInPermanentChoice
```

Параметры:

<ИмяРеквизита> Строковое выражение, содержащее имя реквизита формы подбора значения.

<Значение> Выражение, значение которого заносится в реквизит.

Описание:

Метод УстановитьЗначениеВПодборе, который доступен только в контексте Модуля формы, используется для того, чтобы в окне диалога, открытом для подбора значения, установить значение некоторого реквизита диалога (он может быть затем использован для отображения некоторой информации в информационных элементах диалога подбора значения).

Данный метод следует использовать после вызова метода ОткрытьПодбор.

Пример:

```
Процедура Подбор()
   ГруппаДляВыбора = ТекуцийЭлемент();
   Если ГруппаДляВыбора.ЭтоГруппа() = 0 Тогда
      ГруппаДляВыбора = ГруппаДляВыбора.Родитель;
   КонецЕсли;
   ОткрытьПодбор("Справочник.Товары", "ДляПодбора", КонтПодб);
   // установим в форме подбора реквизит "ГруппаВыбора"
   // используя метод УстановитьЗначениеВПодборе
   УстановитьЗначениеВПодборе("ГруппаВыбора", ГруппаДляВыбора);
```

```
КонецПроцедуры
```
См. также: ОткрытьПодбор, ОбработкаПодбора

ПолучитьЗначениеИзПодбора

Вычислить выражение в контексте открытой формы подбора.

Синтаксис:

```
ПолучитьЗначениеИзПодбора(<Выражение>)
```

Англоязычный синоним:

```
GetValueFromPermanentChoice
```

Параметры:

<Выражение>	Строковое выражение, которое должно содержать запись выражения на встроенном языке. Значение этого выражения вычисляется в контексте открытой формы подбора.

Возвращаемое значение:

Значение вычисленного в контексте формы подбора выражения.

Описание:

Метод ПолучитьЗначениеИзПодбора, который доступен только в контексте Модуля формы, позволяет получить результат выражения, выполненного в контексте формы подбора (т. е. получить любые данные из той формы, например выбранный элемент списка справочника подбора значения).

Данный метод следует использовать только после вызова метода ОткрытьПодбор.

Пример:

```
Процедура ОбработкаПодбора(Выб, Конформы)
   Кол = 0;
   Если ВвестиЧисло(Кол, "Введите количество", 10, 0) = 1 Тогда
      НоваяСтрока();
      ТипРаб = Выб;
      Количество = Кол;
      АктивизироватьСтроку();
      Активизировать("Стоимость", 0);
   КонецЕсли;
   // тип выполненной работы
   ТР = ПолучитьЗначениеИзПодбора("Наименование");
КонецПроцедуры
```

См. также: ОткрытьПодбор, ОбработкаПодбора

Активизировать

Установить курсор на выбранный элемент диалога.

Синтаксис:

```
Активизировать(<ИмяРеквизита>, <Режим>)
```

Англоязычный синоним:

```
Activate
```

Параметры:

<ИмяРеквизита>	Необязательный параметр. Строковое выражение, содержащее имя элемента диалога, который должен быть активизирован. Пустое имя элемента диалога используется для активизации всей формы.
<Режим>	Необязательный параметр. Имеет смысл только для реквизитов многострочной части формы документа. Числовое выражение: 1 — войти в режим редактирования; 0 — не входить в режим редактирования. Значение по умолчанию — 1.

Описание:

Метод Активизировать, который доступен только в контексте Модуля формы, устанавливает курсор для редактирования нужного элемента диалога. Например, если необходимо установить курсор на реквизит многострочной части документа (в форме документа) после выхода из предопределенной процедуры ОбработкаПодбора.

Метод Активизировать может быть вызван из другого (внешнего) модуля, если в нем известен контекст формы, в которой нужно активизировать элемента диалога.

Пример:

См. предыдущий пример.

АктивизироватьОбъект

Установить курсор в списке на выбранном объекте.

Синтаксис:

```
АктивизироватьОбъект(<Объект>)
```

Англоязычный синоним:

```
ActivateObj
```

Параметры:

<Объект>	Выражение, которое может содержать значение элемента справочника или документ или запись журнала расчетов (в зависимости от типа модуля формы, в котором должен быть активизирован элемент диалога).

Описание:

Метод АктивизироватьОбъект, который доступен только в контексте Модуля формы, устанавливает курсор на нужной строке списка диалога, например документ в форме журнала или элемент в форме списка справочника, или запись расчета в форме журнала расчетов.

Пример:

```
АктивизироватьОбъект(ВыбДокумент);
```

Предопределенные процедуры Модуля формы

Описанные в данном разделе системные предопределенные процедуры должны располагаться только в Модулях форм (см. «Виды программных модулей»). К Модулям форм относятся практически все программные модули (исключение составляют Глобальный модуль, Модуль документа, Модуль вида расчета).

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».

ПриОткрытии

Предопределенная процедура при открытии формы.

Синтаксис:

```
ПриОткрытии()
```

Англоязычный синоним:

```
OnOpen
```

Описание:

Вызов предопределенной процедуры ПриОткрытии производится самой системой 1С:Предприятие неявно при интерактивном открытии формы. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю зыпрещено просматривать форму объекта), открытие формы не будет выполнено.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ПриОткрытии()
   ПриЗаписиПерепроводить(1);
КонецПроцедуры
```

См. также: СтатусВозврата

ПриПовторномОткрытии

Предопределенная процедура при повторном открытии формы.

Синтаксис:

```
ПриПовторномОткрытии()
```

Англоязычный синоним:

```
OnReopen
```

Описание:

Вызов предопределенной процедуры ПриПовторномОткрытии производится самой системой 1С:Предприятие неявно при открытии формы, в случае, если открывают уже открытую форму — то есть форма просто активизируется.

Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю запрещено просматривать форму объекта), открытие формы не будет выполнено.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ПриПовторномОткрытии()
   ПриЗаписиПерепроводить(1);
КонецПроцедуры
```

См. также: СтатусВозврата

ПриЗакрытии

Предопределенная процедура при закрытии формы.

Синтаксис:

```
ПриЗакрытии()
```

Англоязычный синоним:

```
OnClose
```

Описание:

Вызов предопределенной процедуры ПриЗакрытии производится самой системой 1С:Предприятие неявно при интерактивном закрытии формы. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если неверно или не полностью заполнены реквизиты объекта), закрытие формы не будет выполнено.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ПриЗакрытии()
```

```
   Если ПоСчетуФактуре.Выбран() = 0 Тогда
      Предупреждение("Укажите Счет-Фактуру", 4);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

ПриВыбореЗакладки

Предопределенная процедура смены закладки формы.

Синтаксис:

```
ПриВыбореЗакладки(<НомерЗакладки>, <ЗначениеЗакладки>)
```

Англоязычный синоним:

```
OnTabCtrlPosChanged
```

Параметры:

<НомерЗакладки>	Числовое значение — номер выбранной закладки формы.
<ЗначениеЗакладки>	Значение выбранной закладки формы.

Описание:

Вызов предопределенной процедуры ПриВыбореЗакладки производится в системе 1С:Предприятие неявно в момент интерактивного выбора пользователем закладки в форме. При вызове процедуры, система подставляет фактические значения параметров, характеризующие выбранную закладку.

Параметры <НомерЗакладки> и <ЗначениеЗакладки> используется в теле процедуры для обработки передаваемого системой события смены закладки формы.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ЗакладкаОбщиеВидимость(ФлагВидимости)
   Форма.Название.Видимость(ФлагВидимости);
   Форма.Наименование.Видимость(ФлагВидимости);
   Форма.Код.Видимость(ФлагВидимости);
   Форма.Статус.Видимость(ФлагВидимости);
   Форма.Адрес.Видимость(ФлагВидимости);
   Форма.Телефон.Видимость(ФлагВидимости);
   Активизировать("Наименование", 0);
КонецПроцедуры

Процедура ЗакладкаЗаметкиВидимость(ФлагВидимости)
   Форма.Заметки.Видимость(ФлагВидимости);
   Активизировать("Заметки", 0);
КонецПроцедуры

Процедура ПриВыбореЗакладки(НомерЗакладки, ЗначениеЗакладки)
   Если ЗначениеЗакладки = 1 Тогда
      Закладка.Заметки.Видимость(0);
      Закладка.Общие.Видимость(1);
   ИначеЕсли ЗначениеЗакладки = 2 Тогда
      Закладка.Общие.Видимость(0);
      Закладка.Заметки.Видимость(1);
   КонецЕсли;
КонецПроцедуры
```

См. также: ИспользоватьЗакладки, Закладки, СтатусВозврата

ПриНачалеВыбораЗначения

Предопределенная процедура при начале выбора значения.

Синтаксис:

```
ПриНачалеВыбораЗначения(<ИдентЭлемДиалога>, <ФлагСтандОбр>)
```

Англоязычный синоним:

```
OnStartValueChoice
```

Параметры:

<ИдентЭлемДиалога>	Строковое значение — идентификатор элемента диалога формы.
<ФлагСтандОбр>	Изначально, при вызове процедуры равен 1, если в теле процедуры значение этого параметра поменять на 0, то стандартный процесс выбора значения не будет происходить.

Описание:

Вызов предопределенной процедуры ПриНачалеВыбораЗначения производится в системе 1С:Предприятие неявно в момент, когда в форме пользователь интерактивно инициировал выбор значения (выбор может быть интерактивно инициирован в немодальном режиме при помощи элемента диалога с кнопкой выбора или клавишей «F4»). При вызове процедуры, система подставляет фактические значения параметров, характеризующие элемент диалога и флаг стандартной обработки.

Замечание. В теле этой процедуры методы ОткрытьФорму и ОткрытьПодбор работают для выбора.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ПриНачалеВыбораЗначения(Элемент, Флаг)
   Перем КонтПодб;
   Если Элемент = "ИдВыбКлиент" Тогда
      Флаг = 0;
      ОткрытьПодбор("Справочник.Клиенты", "ДляПодбора", КонтПодб);
   КонецЕсли;
КонецПроцедуры
```

ОбработкаПодбора

Предопределенная процедура обработки подбора значения.

Синтаксис:

```
ОбработкаПодбора(<ЗначениеПодбора>, <КонтФормы>)
```

Англоязычный синоним:

```
ProcessPermanentChoice
```

Параметры:

<ЗначениеПодбора>	Элемент справочника или документ, передаваемый для обработки.
<КонтФормы>	Контекст той формы, из которой шел подбор.

Описание:

Вызов предопределенной процедуры ОбработкаПодбора производится в системе 1С:Предприятие неявно после нажатия кнопки «Выбрать» в форме подбора значения (см. метод ОткрытьПодбор). В этот момент система подставляет фактическое значение параметра <ЗначениеПодбора>.

Формальный параметр <ЗначениеПодбора> используется в теле процедуры для приема и обработки передаваемого системой значения элемента подбора.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ОбработкаПодбора(Знач, КонтФормы)
   НоваяСтрока();
   Товар = Знач;
   Знач.ИспользоватьДату(ДатаДок);
   Цена = Знач.Цена;
   Сообщить(Знач.Цена);
   Сообщить(Знач.МОЛ.Наименование);
   АктивизироватьСтроку();
КонецПроцедуры
```

См. также: ОткрытьПодбор

ОбработкаВыбораЗначения

Предопределенная процедура обработки выбора значения.

Синтаксис:

```
ОбработкаВыбораЗначения(<ВыбЗнач>, <ИдентЭлемДиалога>, <ФлагСтандОбр>)
```

Англоязычный синоним:

```
ProcessPermanentChoice
```

Параметры:

<ВыбЗнач>	Выбранный элемент справочника, документ или иной объект, передаваемый для обработки.
<ИдентЭлемДиалога>	Идентификатор элемента диалога, которым инициализирован выбор значения.
<ФлагСтандОбр>	Флаг, установка которого в теле процедуры в 0 (ноль) приведет к отмене стандартного присвоения значения.

Описание:

Вызов предопределенной процедуры ОбработкаВыбораЗначения производится в системе 1С:Предприятие неявно, после выбора значения в форм[6] выбора (выбор может быть инициирован в немодальном режиме интерактивно, при помощи элемента диалога с кнопкой выбора). В момент выбора система автоматически подставляет фактическое значение параметра <ВыбЗнач>.

Формальный параметр <ВыбЗнач> используется в теле процедуры для приема и обработки передаваемого системой выбранного значения.

Данная предопределенная процедура может располагаться только в программном модуле формы.

Пример:

```
Процедура ОбработкаВыбораЗначения(Знач, Идент, Флаг)
   ...
КонецПроцедуры
```

ПриВыбореСтроки

Предопределенная процедура при выборе строки списка.

Синтаксис:
ПриВыбореСтроки()
Англоязычный синоним:
OnSelectLine
Описание:
Вызов предопределенной процедуры ПриВыбореСтроки производится в системе 1С:Предприятие при интерактивном выборе строки списка в форме списка справочника, журнала документов, счетов, журнала операций, журнала проводок. Если в данной предопределенной процедуре установить статус возврата — 0 (например, если данному пользователю нельзя вводить новые строки списка справочника), то строка списка не будет выбрана.

Замечание: Режим обработки выбора строки (двойной щелчок мыши или клавиша Enter) предопределенной процедурой ПриВыбореСтроки включается в форме списка справочника, журнала, счетов, журнала операций, журнала проводок при помощи метода Форма.ОбработкаВыбораСтроки(1)

Данная предопределенная процедура может располагаться в Модуле формы списка справочника, журнала, счетов, журнала операций, журнала проводок (см. «Виды программных модулей»).
Пример:
```
Процедура ПриВыбореСтроки()
   Если НазваниеНабораПрав() = "Продавец" Тогда
      Предупреждение("У вас нет права просмотра строки!", 2);
      СтатусВозврата(0);
   КонецЕсли;
КонецПроцедуры
```
См. также: СтатусВозврата, ОбработкаВыбораСтроки

Атрибуты и методы контекста Модуля формы отчета (обработки)

Таблица

Объект агрегатного типа «Таблица».
Синтаксис:
Таблица
Англоязычный синоним:
Table
Описание:
Атрибут Таблица (только для чтения) представляет собой ссылку на объект типа «Таблица». Доступ к данному атрибуту возможен только в контексте Модуля формы отчета или обработки. При настройке формы отчета (обработки), если табличный документ размещен непосредственно в форме (для этого в диалоге, вызываемом пунктом «Свойства формы» меню «Действия» в параметре «Использовать таблицу» выбирается вариант «Пустую» или «Для ввода данных»), то доступ к такому объекту осуществляется через атрибут контекста формы отчета (обработки) Таблица.

Атрибуты и методы объекта «Таблица» описаны в разделе «Работа с таблицами» и позволяют в программном модуле управлять процессом формирования и визуального отображения таблицы в целом, а также изменять свойства визуального отображения отдельных областей таблицы.

В тексте программного модуля через точку после имени атрибута «Таблица» можно записывать адреса областей таблицы, а далее через точку можно вызывать методы управления свойствами этих областей.
Пример:
```
Таблица.ИсходнаяТаблица("price");
Таблица.Вывести();
Таблица.ТолькоПросмотр(0);
Таблица.Показать("Каталог", "Catalog.mxl");
Таблица.ТолькоПросмотр(1);
```
Пример:
```
ВыбОбласть = Таблица.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ЦветФона(34, 126, 211);
```
См. также: разд. «Работа с таблицами»

<ИмяОбласти>

Возвращает или задает значение, записанное в именованной области таблицы в режиме ввода данных.
Синтаксис:
<ИмяОбласти> Имя области таблицы, как она задана в табличном документе.
Описание:

Атрибут <ИмяОбласти> позволяет обращаться к значениям, записанным в поименованных областях таблицы в режиме ввода данных. Для обращения к значению конкретной области следует указать имя этой области, заданный в конфигураторе.

Данный атрибут существует и доступен в форме, если при конфигурировании данной формы в свойствах формы выбраны опция «Использовать таблицу» — «Для ввода данных». Установка данной опции в свойствах формы означает наличие у данной формы таблицы в режиме ввода данных. В процессе конфигурирования в таблице можно задать любое число выделенных и поименованных областей.

Доступ к данному атрибуту возможен только в контексте Модуля формы отчета или обработки.

Замечание: Данный атрибут работает на чтение только в том случае, если в качестве области таблицы помечена одна единственная ячейка.

Пример:

* В диалоге формы есть элемент диалога ВыбКод. В форме использована таблица в режиме ввода данных. Одна из ячеек таблицы помечена как область с именем «ВыбранныйКлиент». В этом случае в модуле формы можно задать значение данной области.

```
Клн = СоздатьОбъект("Справочник.Клиенты");
Клн.НайтиПоКоду(ВыбКод, 0);
ВыбранныйКлиент = Клн.ТекущийЭлемент();
```

См. также: разд. «Работа с таблицами»

РасположениеФайла

Определить, где располагается данный внешний отчет.

Синтаксис:

```
РасположениеФайла(<Путь>, <Имя>)
```

Англоязычный синоним:

```
FilePath
```

Параметры:

<Путь>	Идентификатор переменной, куда метод возвращает путь к файлу, где располагается данный внешний отчет.
<Имя>	Идентификатор переменной, куда метод возвращает имя файла, где располагается данный внешний отчет.

Возвращаемое значение:

Строковое значение полного имени (вместе с путем) файла, где располагается данный внешний отчет.

Описание:

Метод РасположениеФайла позволяет узнать, где располагается данный внешний отчет.

Замечание. Данный метод следует использовать только в модуле формы внешнего отчета.

Пример:

```
Имя = "";
Путь = "";
ПолноеИмя = РасположениеФайла(Путь, Имя);
```

Предопределенные процедуры модуля формы отчета (обработки)

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 открытии формы отчета (обработки) или при восстановлении сохраненной настройки отчета (обработки).

Синтаксис:

```
ВводНового()
```

Англоязычный синоним:

```
InputNew
```

Описание:

При работе с формой отчета у пользователя существует возможность сохранять и восстанавливать настройки отчета — содержимое реквизитов диалога формы. Предопределенная процедура ВводНового вызывается и в момент открытия формы и в момент восстановления значений настройки. Это позволяет контролировать содержимое реквизитов диалога во всех случаях их изменения системой.

Пример:

```
Процедура ВводНового()
   ДатаНач = РабочаяДата();
   ДатаКон = РабочаяДата();
КонецПроцедуры
```

См. также: СтатусВоз врата

ПриОткрытии

Предопределенная процедура при открытии формы отчета (обработки).

Синтаксис:

ПриОткрытии(<ФлагЧтенияНастройки>)

Англоязычный синоним:

OnOpen

Параметры:

<ФлагЧтенияНастройки>	Числовое значение — признак считывания сохраненной настройки отчета (обработки). Может принимать значения: 1 — при открытии формы была восстановлена последняя сохраненная настройка отчета (обработки); 0 — при открытии формы настройка не восстановлена.

Описание:

Форма любого отчета или обработки в системе 1С:Предприятие обязательно содержит экранный диалог. В диалоге могут размещаться элементы для задания различных параметров построения отчета или выполнения обработки. При Использовании отчета (обработки) набор параметров, использованных при формировании отчета или выполнения обработки, можно запомнить, а при следующем использовании этого же отчета (обработки) — восстановить. Для выполнения этих операций существуют команды в меню «Действия» системы 1С:Предприятие.

Набор параметров формирования отчета или выполнения обработки называется настройкой отчета (обработки). Последняя сохраненная настройка автоматически восстанавливается системой 1С:Предприятие, при вызове отчета (обработки).

Вызов предопределенной процедуры ПриОткрытии производится самой системой 1С:Предприятие неявно при интерактивном открытии отчета (обработки). Параметр <ФлагЧтенияНастройки> позволяет определить, была ли при открытии отчета (обработки) восстановлена сохраненная настройка.

Если в данной предопределенной процедуре установить статус возврата 0 (например, если данному пользователю запрещено формировать выбранный отчет), открытие формы не будет выполнено.

Пример:

```
Процедура ПриОткрытии(ФлагСохрНастр)
   Если ФлагСохрНастр = 0 Тогда
      ВыбВалюта = Константа.ОснВалюта;
   КонецЕсли;
КонецПроцедуры
```

См. также: СтатусВозврата

Глава 31
Работа с Таблицами

Для работы с табличными документами в системе используется специальный агрегатный тип данных «Таблица». Основным назначением табличного документа в системе 1С:Предприятие является создание печатных форм отчетов и первичных документов.

Печатные формы формируются при помощи встроенного языка с использованием агрегатного объекта «Таблица», создаваемого функцией `СоздатьОбъект`. Наиболее типичным способом заполнения табличного документа является включение в него секций. Для этого используется исходный табличный документ, который может располагаться в той же форме, в тексте программного модуля которого создан агрегатный объект «Таблица». Кроме того, исходная таблица может располагаться и в общих таблицах конфигурации и во внешнем файле. Назначение исходного табличного документа выполняется методом объекта «Таблица» — `ИсходнаяТаблица`.

Исходная таблица является заготовкой, содержащей текстовые фрагменты отчета, рамки, рисунки и т. д. Исходная таблица может быть включена в создаваемый отчет целиком. Однако часто необходимо включать в отчет отдельные фрагменты заготовки в определенной последовательности. Например, при печати накладной, нужно один раз вывести шапку и много раз вывести строку документа. Для этого части исходной таблицы выделяются в виде фрагментов-секций. Метод `ВывестиСекцию` позволяет включить выделенный фрагмент исходной таблицы в формируемый табличный документ.

В ячейках исходной таблицы может размещаться обычный текст или выражение встроенного языка 1С:Предприятия. При включении в табличный документ секции исходной таблицы автоматически происходит вычисление всех выражений и в табличный документ уже включается секция, содержащая результаты этих вычислений. Таким образом, происходит заполнение включаемых секций конкретными данными. Описанный способ, с одной стороны, позволяет наиболее наглядным образом визуально в исходной таблице спроектировать внешний вид и содержание отчета, а с другой, позволяет достаточно просто включать подготовленные фрагменты в заполняемый табличный документ.

Приведем пример формирования таким способом печатной формы документа.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("ПечатьСчета");
Таб.ВывестиСекцию("Шапка");
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
   Таб.ВывестиСекцию("Строка");
КонецЦикла;
Таб.Показать();
```

В приведенном примере в ячейках секций "`Шапка`" и "`Строка`" исходной таблицы располагаются выражения выдающие различные реквизиты документа.

Существует другой способ заполнения данными включаемых секций. Этот способ отличается от описанного тем, что в секциях ячеек не задаются выражения. Заполнение секции данными описывается прямо в алгоритме программного модуля. Для этого секция сначала выбирается из исходной таблицы, затем в ней заполняются значениями ячейки, в которых должны выводиться данные, а затем секция включается в табличный документ. Для получения секции из исходной таблицы используется метод `ПолучитьСекцию`. Полученная секция запоминается в переменной как специальный объект типа «СекцияТаблицы». Для того, чтобы в этом объекте можно было заполнять отдельные ячейки им нужно предварительно задать имена в исходной таблице. У объекта «СекцияТаблицы» отдельные именованные ячейки являются атрибутами, имеющими тип «ОбластьТаблицы» (область может содержать и несколько ячеек). Заполнение ячеек выполняется присвоением атрибуту "`Текст`" объекта «ОбластьТаблицы» необходимых значений.

Приведем пример формирования табличного документа этим способом.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("ПечатьСчета");
Шапка = Таб.ПолучитьСекцию("Шапка");
Шапка.Клиент.Текст = Контрагент;
Шапка = Таб.ВывестиСекцию(Шапка);
......
ВыбратьСтроки();
Пока ПолучитьСтроку () = 1 Цикл
   Строка = Таб.ПолучитьСекцию("Строка");
   Строка.Товар.Текст = Товар;
   ......
   Строка = Таб.ВывестиСекцию(Строка);
КонецЦикла;
Таб.Показать();
```

Этот способ может применяться в отдельных случаях для ускорения заполнения табличного документа, если для заполнения данными используются достаточно сложные выражения. Кроме того, этот способ позволяет манипулировать при включении секций различными свойствами ячеек. Например, можно для отдельных ячеек устанавливать жирный шрифт.

Кроме описанных способов существует способ заполнения табличного документа путем непосредственного обращения к ячейкам таблицы без использования в качестве заготовки исходной таблицы. Для этого используется метод Область, позволяющий по координатам строки и столбца исходной таблицы получить объект типа «ОбластьТаблицы». Атрибуты и методы этого объекта позволяют задать содержимое и элементы оформления ячеек (текст, цвета, рамки и т. д.).

Приведем фрагмент примера печати документа этим способом.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Область(2, 3).Текст = Контрагент;
......
ВыбратьСтроки();
Пока ПолучитьСтроку() = 1 Цикл
   Таб.Область(НомерСтроки + 3, 3).Текст = Товар;
   ......
КонецЦикла;
Таб.Показать();
```

Этот способ используется в основном для создания универсальных отчетов, для которых невозможно определить заранее их внешний вид. Заметим, что такой способ является наиболее трудоемким с точки зрения написания алгоритма.

Во всех описанных способах основным содержанием ячеек таблицы является текст. При заполнении ячеек все данные преобразуются к текстовому виду. Кроме того, у ячейки может заполняться значение расшифровки. Это дополнительная возможность, которая никак не влияет на внешний вид таблицы при просмотре и печати, а используется для создания связанных отчетов и открытия различных объектов, отображаемых в отчете. При просмотре сформированного отчета в режиме «только просмотр», при подведении курсора к ячейке, у которой заполнено значение расшифровки, курсор принимает форму крестика с лупой и при двойном щелчке мышью или нажатии клавиши <Enter> выполняется отработка значения расшифровки.

Стандартный механизм расшифровки позволяет открыть объект (документ, элемент справочника, счет). Кроме того, использование специальной предопределенной процедуры позволяет определить в модуле формы специальные действия по отработки расшифровки, например построение детализирующего отчета.

Заполнение значения расшифровки при включении секций из исходной таблицы выполняется автоматическим вычислением значения выражения указанного в поле «Расшифровка» закладки «Текст» свойств ячейки. При использовании двух других описанных способов значение расшифровки может быть заполнено с помощью метода `Расшифровка` объекта типа «ОбластьТаблицы».

Заметим, что указанием "#" в поле «Расшифровка» закладки «Текст» или параметра метода `Расшифровка` можно включить режим, при котором расшифровка будет действовать на всю строку таблицы, что позволит избежать Дублирования значений для нескольких ячеек и, тем самым, избежать излиш-иих затрат времени при формировании отчета на вычисление выражений расшифровки.

Описанный выше способ заполнения табличного документа созданного функцией `СоздатьОбъект` открывает табличный документ в отдельном окне при использовании метода `Показать`. Кроме того, при настройке формы отчета (обработки), существует возможность табличный документ разместить непосредственно в форме. Для этого в диалоге, вызываемом пунктом «Свойства формы» меню «Действия» в параметре «Использовать таблицу» выбирается вариант «Пустую». В этом случае при открытии формы непосредственно в ней размещается табличный документ. Он может заполняться так же, как и табличный документ, созданный функцией `СоздатьОбъект`. Для доступа к такой таблице используется атрибут контекста формы отчета (обработки) Таблица.

Кроме описанного использования табличного документа для вывода сформированных печатных форм, существует возможность использования табличного документа в специальном режиме для ввода данных. Этот режим доступен только в формах отчета (обработки). Данный режим использования табличного документа включается через свойства формы (вызов свойств формы выполняется из меню «Действия» пунктом «Свойства формы» в параметре «Использовать таблицу» выбирается вариант «Для ввода данных»).

Режим ввода данных позволяет совместить в одной форме отчета (обработки) диалог формы вместе с табличным документом или вовсе заменить диалог табличным документом. Этот режим предполагает обязательное наличие созданной в форме отчета (обработки) таблицы, которая выбирается в свойствах формы, как используемая для ввода данных. В этом режиме в свойствах ячеек табличного документа появляется дополнительная закладка, на которой определяется тип данных хранимых ячейкой, аналогично элементу диалога формы. Для ячеек, в которых предполагается вводить данные, в закладке «Текст» выключается признак «Защита». При открытии формы с установленным свойством использование таблицы для ввода данных, указанная таблица размещается в окне формы и пользователю предоставляется возможность интерактивно вводить значения ячеек, предназначенных для ввода, аналогично вводу в поля диалога.

Работа в режиме ввода данных имеет ряд особенностей с точки зрения обращения к таблице средствами встроенного языка. Прежде всего, доступ к используемой для ввода данных таблице в модуле формы отчета (обработки) осуществляется с помощью ключевого слова `"Таблица"`. Кроме того, в модуле формы возможно непосредственное обращение к значениям именованных ячеек таблицы по их именам. При обращении к области таблицы, используемой в режиме ввода данных, кроме обычных атрибутов используется атрибут `"Значение"`, также предоставляющий доступ к значению ячейки. При этом тип значения определяется типом, выбранным в закладке свойств ячейки «Данные».

Для таблицы, используемой для ввода, в свойствах ячеек данных задаются-формулы. Для ячеек, предназначенных для ввода, формула выполняется после ввода значений, как у реквизитов диалога и используется обычно для вызова

процедуры заполнения других ячеек. У ячеек имеющих признак «Защита» формула является выражением и вычисляет собственно значение ячейки как у элементов диалога типа «Текст».

Применение табличного документа для ввода данных используется в специальных случаях, например, когда необходимо ввести большое количество данных или необходимо, чтобы форма ввода была оформлена с использованием разнообразных шрифтов, рамок, картинок и т. п.

Контекст работы с таблицами

Объект типа «Таблица» создается функцией `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

Для создания объекта типа «Таблица» в качестве параметра функции `СоздатьОбъект` передается ключевое слово `"Таблица"`.

Англоязычный синоним ключевого слова `Таблица` — `Table`.

Пример:

```
// Подготовка к заполнению выходных форм
Таб = СоздатьОбъект("Таблица");
Пока Запрос.Группировка("Сотр") = 1 Цикл
   // Заполнение полей Сотр
   Таб, ВывестиСекцию("РЛ<");
   Пока Запрос.Группировка("Расч") = 1 Цикл
      // Заполнение полей Расч
      Таб.ВывестиСекцию("ОТ");
   КонецЦикла;
   Таб.ВывестиСекцию("РЛ>");
КонецЦикла;
// Вывод заполненной формы
Таб.Опции(0, 0, 0, 0);
Таб.ТолькоПросмотр(1);
Таб.Показать("Результат");
```

При настройке формы отчета (обработки), если табличный документ размещен непосредственно в форме (для этого в диалоге, вызываемом пунктом «Свойства формы» меню «Действия» в параметре «Использовать таблицу» выбирается вариант «Пустую»), то доступ к такому объекту осуществляется через атрибут контекста формы отчета (обработки) `Таблица`.

Атрибуты таблиц

ТекущийОбъект

Значение встроенного объекта таблицы.

Синтаксис:

`ТекущийОбъект`

Англоязычный синоним:

`CurrentObj`

Описание:

Атрибут (только для чтения) `ТекущийОбъект` содержит значение встроенного объекта таблицы типа «Картинка», «Диаграмма» или «OLE объект». Он доступен только в выражении (и в теле процедур, которые вызываются в выражении), которое записывается в конфигураторе в свойствах встроенного объекта таблицы типа «Картинка», «Диаграмма» или «OLE объект» («Свойства» — «Дополнительныо» — «Текст»). Этот атрибут позволяет использовать в программном модуле полученное значение объекта для манипуляции с ним. Например, для взаимодействия с внешним приложением через механизм OLE Automation.

Пример:

* Допустим, в таблицу для построения диаграммы внедрен OLE-объект «MicrosoftGraf97». В свойствах этого объекта («Свойства» — «Дополнительные» — «Текст») в конфигураторе записано выражение:

```
ПостроитьГрафик(Таб.ТекущийОбъект, Запрос)
```

в программном модуле, в процедуре `ПостроитьГрафик`, можно записать алгоритм построения графика, непосредственно управляя внешним приложением MicrosoftGraf97.

```
Процедура ПостроитьГрафик(График, Запрос)
   Datasheet = График.Application.Datasheet;
   XCounter = 2;
   YCounter = 2;
   Пока Запрос.Группировка("Клиент") = 1 Цикл
      Если Запрос.Клиент.ЭтоГруппа() = 1 Тогда
         Продолжить;
      КонецЕсли;
      range = Datasheet.Cells(1, XCounter);
      range.Value = Запрос.Клиент.Наименование;
      Пока Запрос.Группировка("Товар") = 1 Цикл
         range = Datasheet.Cells(YCounter, 1);
```

```
            range.Value = Запрос.Товар.Наименование;
            range = Datasheet.Cells(YCounter, XCounter);
            range.Value = Запрос.ПродСум;
            YCounter = YCounter+1;
        КонецЦикла;
        YCounter = 2;
        XCounter = XCounter+1;
    КонецЦикла;
КонецПроцедуры
```

Замечание. Следует отметить, что в процессе конфигурирования, когда в таблицу внедряется объект, он может быть визуально предварительно полностью настроен. В данном примере, при внедрении «MicrosoftGraf97» ему установили все параметры требуемой диаграммы: размеры, вид, необходимые надписи и т. п. Атрибут `ТекущийОбъект` содержит объект таким, как он настроен в конфигураторе, поэтому в режиме исполнения ему достаточно только передать актуальные данные для работы.

См. также: «Работа с Диаграммами», «Связь с внешними приложениями посредством механизмов OLE Automation»

Методы таблиц

ИсходнаяТаблица

Переназначить исходную таблицу.

Синтаксис:

`ИсходнаяТаблица(<Строка>)`

Англоязычный синоним:

`SourceTable`

Параметры:

`<Строка>`	Строковое выражение, содержащее имя исходной таблицы формы, определенное в конфигураторе, или имя файла, содержащего таблицу.

Описание:

Метод `ИсходнаяТаблица` переназначает в качестве исходной таблицы-шаблона одну из таблиц той формы, в программном модуле которой запущена данная процедура. Имя таблицы сначала ищется в форме модуля, потом в общих таблицах. Если такой таблицы нет, то переданное имя будет рассматриваться как имя файла, содержащего данную таблицу.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("price");
Таб.Вывести();
Таб.ТолькоПросмотр(0);
Таб.Показать("Каталог", "Catalog.mxl");
Таб.ТолькоПросмотр(1);
```

ИспользоватьФормат

Устанавливает стандартный формат по умолчанию для вывода всех выражений выводимых секций таблицы.

Синтаксис:

`Использоватьформат(<СтрокаФормата>)`

Англоязычный синоним:

`DefaultFormat`

Параметры:

`<СтрокаФормата>`	Необязательный параметр. Строковое выражение, содержащее форматную строку (см. `Формат`).

Возвращаемое значение:

Строковое значение, содержащее текущую форматную строку по умолчанию для таблицы (на момент до исполнения метода).

Описание:

Метод `Использоватьформат` устанавливает стандартный формат по умолчанию для вывода всех выражений выводимых секций таблицы. В ячейках таблицы, при выводе которых требуется формат, отличный от установленного методом `ИспользоватьФормат`, должен быть установлен формат явным образом. Форматная строка записывается через символ "#" после выражения, заданного для ячейки. Если выражение, заданное для ячейки просто завершается символом "#", то будет использоваться системный формат по умолчанию.

Пример:

```
Таб.ИспользоватьФормат("415.2");
```

Открыть

Открыть таблицу из файла.

Синтаксис:

Открыть(<ИмяФайла>)

Англоязычный синоним:

Open

Параметры:

<ИмяФайла> Строковое выражение с именем файла.

Описание:

Метод Открыть открывает таблицу из файла с именем <ИмяФайла>.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("\v7\db\src.mox");
Таб.Показать();
```

Вывести

Перенести всю исходную таблицу-шаблон в результирующую таблицу.

Синтаксис:

Вывести()

Англоязычный синоним:

Put

Описание:

Метод Вывести целиком переносит исходную таблицу-шаблон в результирующую таблицу.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.ИсходнаяТаблица("catalog");
Таб.Вывести();
Таб.ТолькоПросмотр(0);
Таб.Показать("Каталог", "catalog.mxl");
Таб.ТолькоПросмотр(1);
```

ПолучитьСекцию

Получить значение секции исходной таблицы.

Синтаксис:

ПолучитьСекцию(<ИмяСекции>)

Англоязычный синоним:

GetSection

Параметры:

<ИмяСекции> Выражение типа строка, задающее имя секции.

Возвращаемое значение:

Объект типа секция.

Описание:

Метод ПолучитьСекцию возвращает именованную секцию исходной таблицы-шаблона. При получении секции, ячейки секции, имеющие тип «Шаблон» и «Выражение» будут заполнены соответствующими данными. Имя секции задается строковым выражением следующего формата:

ИдентификаторСекции1[<|>|-][|ИдентификаторСекции2[<|>|-]]

Символы "<" , ">" , "-" после идентификатора секции указывают на то, что выбирается только часть секции:

<	заголовочная часть (с начала секции до начала вложенной секции).
>	подвальная часть (с конца вложенной секции до конца секции).
—	средняя часть (собственно вложенная секция).

В выражении <ИмяСекции> можно задавать имена двух секций, разделенных знаком "|". При этом будет получена область исходной таблицы, являющаяся пересечением первой и второй указанных секций. При этом одна секция может быть горизонтальной (состоять из строк), а другая — вертикальной (состоять из колонок). В результате получится прямоугольная область таблицы.

Пример:

```
//Выводим секцию таблицы
Секция = Таб.ПолучитьСекцию("Документ<|ДокументВерт<");
Таб.ВывестиСекцию(Секция);
```

ВывестиСекцию

Перенести секцию исходной таблицы-шаблона в результирующую таблицу.

Синтаксис:

ВывестиСекцию(<Секция>)

Англоязычный синоним:

PutSection

Параметры:

<Секция> Выражение типа строка, задающее имя выводимой секции, или значение типа секция, полученное при помощи метода ПолучитьСекцию.

Описание:

Метод ВывестиСекцию выполняет перенос именованной секции из исходной таблицы-шаблона в результирующую таблицу. Имя секции задается строковым выражением следующего формата:

ИдентификаторСекции1[<|>|-][|ИдентификаторСекции2[<|>|-]]

Символы "<" , ">" , "-" после идентификатора секции указывают на то, что выбирается только часть секции:

<	заголовочная часть (с начала секции до начала вложенной секции).
>	подвальная часть (с конца вложенной секции до конца секции).
—	средняя часть (собственно вложенная секция).

В выражении <Секция> можно задавать имена двух секций, разделенных знаком "|". При этом перенесена будет область исходной таблицы, являющаяся пересечением первой и второй указанных секций. При этом одна секция может быть горизонтальной (состоять из строк), а другая — вертикальной (состоять из колонок). В результате перенесется прямоугольная область таблицы.

Если в качестве параметра метода задано строковое значение имени секции, то при переносе ячейки таблицы, имеющие тип «Шаблон» и «Выражение», будут заполнены соответствующими данными.

Метод ВывестиСекцию помещает новую секцию со следующей строки вслед за последней выведенной секцией, начиная с первой колонки.

Пример:

```
//Выводим заглавие таблицы
Таб.ВывестиСекцию("Документ<|ДокументВерт<");
Пока ЗапросКат.Группировка("Ктг") = 1 Цикл
   Таб.ПрисоединитьСекцию("Документ<|Осн2");
КонецЦикла;
Таб.ПрисоединитьСекцию("Документ<|ДокументВерт>");

//Выводим колонтитул таблицы
Таб.ВывестиСекцию("КолонТитул|ДокументВерт<");
Пока ЗапросКат.Группировка("Ктг") = 1 Цикл
   Таб.ПрисоединитьСекцию("КолонТитул|Осн2");
КонецЦикла;
Таб.ПрисоединитьСекцию("КолонТитул|ДокументВерт>");
Продолжать = 1;
Пока Запрос.Группировка("Пдр") = 1 Цикл
   // Заполнение полей
   Пдр Таб.ВывестиСекцию("Осн1|ДокументВерт<");
   Далее = 1;
   Пока Продолжать = 1 Цикл
      // Заполнение полей Ктг
      СлКат = ЗапросКат.Группировка("Ктг");
      Если Далее = 1 Тогда
         ОК = Запрос.Группировка("Ктг");
      КонецЕсли;
      Если СлКат = 0 Тогда
         Прервать;
      КонецЕсли;
      Если ЗапросКат.Ктг = Запрос.Ктг Тогда
         Таб.ПрисоединитьСекцию("Осн1|Осн2");
         Далее = 1;
      Иначе
         Таб.ПрисоединитьСекцию("Осн1|Пусто");
         Далее = 0;
      КонецЕсли;
   КонецЦикла;
   Таб.ПрисоединитьСекцию("Осн1|ДокументВерт>");
КонецЦикла;
```

ПрисоединитьСекцию

Присоединить секцию исходной таблицы-шаблона к результирующей таблице.

Синтаксис:

ПрисоединитьСекцию(<Секция>)

Англоязычный синоним:

AttachSection

Параметры:

<Секция> Выражение типа строка, задающее имя выводимой секции, или значение типа секция, полученное при помощи метода ПолучитьСекцию.

Возвращаемое значение: Нет.

Описание:

Метод ПрисоединитьСекцию выполняет присоединение именованной секции из исходной таблицы-шаблона к результирующей таблице. Имя секции задается строковым выражением следующего формата:

ИдентификаторСекции1[<|>|-][|ИдентификаторСекции2[<|>|-]]

Символы "<" , ">" , "-" после идентификатора секции указывают на то, что выбирается только часть секции:

<	заголовочная часть (с начала секции до начала вложенной секции).
>	подвальная часть (с конца вложенной секции до конца секции).
—	средняя часть (собственно вложенная секция).

В выражении <Секция> можно задавать имена двух секций, разделенных знаком "|". При этом перенесена будет область исходной таблицы, являющаяся пересечением первой и второй указанных секций. При этом одна секция может быть горизонтальной (состоять из строк), а другая — вертикальной (состоять из колонок). В результате перенесется прямоугольная область таблицы.

Если в качестве параметра метода задано строковое значение имени секции, то при переносе ячейки таблицы, имеющие тип «Шаблон» и «Выражение», будут заполнены соответствующими данными.

Метод ПрисоединитьСекцию помещает новую секцию в следующей колонке, правее последней ранее выведенной секции. При этом секция, передаваемая в качестве параметра метода ПрисоединитьСекцию, должна иметь прямоугольную форму, т. е. задаваться как пересечение горизонтальных и вертикальных секций.

Пример:

См. предыдущий пример.

НоваяСтраница

Вставить в результирующую таблицу разделитель страниц.

Синтаксис:

НоваяСтраница(<Ном>)

Англоязычный синоним:

NewPage

Параметры:

<Ном>	Необязательный параметр. Номер строки, после которой начинать новую страницу. Если параметр не указан, то новая страница вставляется по текущей высоте таблицы.

Описание:

Процедура НоваяСтраница вставляет в результирующую таблицу разделитель страниц.

Пример:

```
Таб.НоваяСтраница();
```

НоваяКолонка

Вставить в результирующую таблицу разделитель колонок.

Синтаксис:

НоваяКолонка(<Ном>)

Англоязычный синоним:

NewColumn

Параметры:

<Ном>	Необязательный параметр. Номер столбца, после которого начинать новую колонку. Если параметр не указан, то новая колонка вставляется по текущей ширине таблицы.

Описание:

Метод НоваяКолонка вставляет в результирующую таблицу разделитель колонок.

Пример:

```
Таб.НоваяКолонка();
```

ШиринаТаблицы

Определить текущую ширину результирующей таблицы.

Синтаксис:

ШиринаТаблицы()

Англоязычный синоним:

TableWidth

Возвращаемое значение:

Числовое значение, содержащее количество столбцов в результирующей таблице.

Описание:

Метод ШиринаТаблицы позволяет определить текущее количество столбцов в результирующей таблице.

Пример:

```
Шир = Таб.ШиринаТаблицы();
```

ВысотаТаблицы

Определить текущую высоту результирующей таблицы.

Синтаксис:

ВысотаТаблицы()

Англоязычный синоним:

TableHeight

Возвращаемое значение:

Числовое значение, содержащее количество строк в результирующей таблице.

Описание:

Метод ВысотаТаблицы позволяет определить текущее количество строк в результирующей таблице.

Пример:

```
Выс = Таб.ВысотаТаблицы();
```

ШиринаСекции

Определить ширину секции таблицы-шаблона.

Синтаксис:

```
ШиринаСекции(<ИмяСекции>)
```

Англоязычный синоним:

```
SectionWidth
```

Параметры:

<ИмяСекции>	Строковое выражение — название секции таблицы.

Возвращаемое значение:

Числовое значение, содержащее количество столбцов в секции таблицы-шаблона.

Описание:

Метод ШиринаСекции позволяет определить количество столбцов в секции таблицы-шаблона.

Пример:

```
ШирС = Таб.ШиринаСекции("БоковикОтчета");
```

ВысотаСекции

Определить высоту секции таблицы-шаблона.

Синтаксис:

```
ВысотаСекции(<ИмяСекции>)
```

Англоязычный синоним:

```
SectionHeight
```

Параметры:

<ИмяСекции>	Строковое выражение — название секции таблицы.

Возвращаемое значение:

Числовое значение, содержащее количество строк в секции таблицы-шаблона.

Описание:

Метод ВысотаСекции позволяет определить количество столбцов в секции таблицы-шаблона.

Пример:

```
ВысС = Таб.ВысотаСекции("ШапкаОтчета");
```

ТолькоПросмотр

Установить режим редактирования таблицы.

Синтаксис:

```
ТолькоПросмотр(<Режим>)
```

Англоязычный синоним:

```
Readonly
```

Параметры:

<Режим>	Необязательный параметр. Числовое выражение — режим редактирования: 1 — только просмотр, 0 — допускается редактирование.

Возвращаемое значение:

Текущее числовое значение режима редактирования таблицы (на момент до исполнения метода).

Описание:

Метод ТолькоПросмотр устанавливает флаг возможности редактирования таблицы в окне. Данный метод должен вызываться до вызова метода Показать. По умолчанию, для табличных документов устанавливается режим с возможностью редактирования.

Замечание: Режим «только просмотр» позволяет воспринимать сгруппированные ячейки таблицы как единое целое и использовать фиксацию шапки и боковика таблицы при просмотре. Таким образом данный режим рекомендуется для отчетов, которые предназначены в основном для просмотра и печати.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("catalog.txt");
Таб.ТолькоПросмотр(0);
Таб.Показать("Редактирование Прайс-листа", "price.txt");
```

Очистить

Очищает содержимое табличного документа.

Синтаксис:

```
Очистить()
```

Англоязычный синоним:

```
Clear
```

Описание:

Метод Очистить очищает текущее содержимое-табличного документа. Его использование позволяет заново заполнить содержимое табличного документа уже открытого в окне методом Показать. Использование данного метода имеет смысл, если сам объект типа «Таблица» не уничтожался после первого заполнения.

Пример:

* Модуль отчета позволяет выводить отчет заново в уже открытое окно.

```
Перем Таб;
Процедура Сформировать()
   Таб.Очистить();
   Таб.ВывестиСекцию("Отчет");
   Таб.Показать();
КонецПроцедуры
Таб=СоздатьОбъект("Таблица");
```

Показать

Открыть окно табличного документа.

Синтаксис:

```
Показать([<Заголовок>], [<ИмяФайла>], [<Активизировать>])
```

Англоязычный синоним:

```
Show
```

Параметры:

<Заголовок>	Необязательный параметр. Строковое выражение — заголовок окна табличного документа.
<ИмяФайла>	Необязательный параметр. Строковое выражение — имя файла для сохранения табличного документа.
<Активизировать>	Необязательный параметр. Числовое выражение — признак активизации. Может принимать значения: 1 — активизировать окно табличного документа; 0 — не активизировать окно табличного документа; -1 (минус единица) — закрыть окно если оно открыто. Значение по умолчанию: 1.

Описание:

Метод Показать открывает окно с табличным документом для просмотра и редактирования.

Параметр <Заголовок> позволяет задать заголовок окна, содержащего табличнй документ. Если параметр не указан, в заголовке будет выдаваться слово «Таблица».

Если указан параметр <ИмяФайла>, то при закрытии окна табличного документа система 1С:Предприятие будет предлагать сохранить документ в файл с указанным именем. Если файла с именем <ИмяФайла> не существует, будет создан новый файл с таким именем.

Если параметр <ИмяФайла> опущен или имеет пустое значение, то при закрытии окна табличного документа система 1С:Предприятие не будет предлагать сохранить данные в файл. Это имеет смысл для документов, которые формируются только для просмотра или печати, и их не обязательно записывать в файл. Вместе с этим, пользователь в любом случае может записать табличный документ в файл, используя команды «Сохранить» и «Сохранить как» из меню «Файл» главного меню системы 1С:Предприятие.

Параметр <Активизировать> позволяет регулировать активизацию окна табличного документа при вызове метода Показать. Если значение параметра — 0 , то окно открывается, но не становится активным.

Пример:

* Создается объект типа «Таблица» для работы с табличными документами, Открывается табличный документ, находящийся в файле SRC.MOX .

```
// Для работы с табличными документами создаем
// объект типа "Таблица"
Таб = СоздатьОбъект("Таблица");
// Открываем табличный документ из файла
Таб.Открыть("\v7\db\src.mox");
// Показ табличного документа
Таб.Показать();
```

Защита

Защитить таблицу от изменений.

Синтаксис:

```
Защита(<Флаг>)
```

Англоязычный синоним:

```
Protection
```

Параметры:

<Флаг>	Необязательный параметр. Числовое выражение — флаг защиты: 1 — установить защиту; 0 — снять защиту. Значение по умолчанию — 1.

Возвращаемое значение:

Текущее значение флага защиты таблицы (на момент до исполнения метода).

Описание:

Метод Защита предназначен для установки режима полной защиты таблицы от редактирования и копирования (в том числе через буфер обмена). Если не использовать метод Защита, то таблица выводится в незащищенном режиме.

Пример:

```
Таб.Защита(1);
```

Записать

Записать таблицу в файл.

Синтаксис:

```
Записать(<ИмяФайла>, <ТипФайла>)
```

Англоязычный синоним:

```
Write
```

Параметры:

<ИмяФайла>	Строковое выражение с именем файла.
<ТипФайла>	Необязательный параметр. Числовое или строковое выражение, определяющее тип файла. • отсутствует, 0 или "MXL" — формат 1С; • 1 или "XLS" — формат MS Excel • 2 или "HTM" или "HTML" — формат HTML; • 3 или "TXT" — формат TXT.

Описание:

Метод Записать записывает таблицу в файл с именем <ИмяФайла>.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
Таб.Записать("tabl_2.mxl");
Таб.Записать("tabl_2", "XLS");
```

ЗначениеТекущейЯчейки

Возвращает вычисленное значение, записанное в ячейку таблицы.

Синтаксис:

```
ЗначениеТекущейЯчейки(<Адрес>)
```

Англоязычный синоним:

```
ValueOfCurrentCell
```

Параметры:

<Адрес>	Необязательный параметр. Идентификатор переменной, куда система возвратит адрес текущей ячейки в формате «RnCn».

Возвращаемое значение:

Значение, записанное в ячейку таблицы.

Описание:

Метод ЗначениеТекущейЯчейки возвращает значение, записанное в ячейку таблицы путем вычисления выражения, заданного в конфигураторе («Свойства» ячейки — закладка «Текст» — поле «Значение»).

Замечание. Для того, чтобы иметь доступ к этому методу, необходимо, чтобы переменная, содержащая таблицу, была объявлена как переменная модуля, а не локальная переменная процедуры. В этом случае во всех процедурах программного модуля вы сможете использовать данный метод и, на основании значения текущей ячейки, формировать, например, более подробные отчеты.

Пример:

```
Док = Таб.ЗначениеТекущейЯчейки();
```

См. также: ЗначениеВСтрокуВнутр, ЗначениеИзСтрокиВнутр

Область

Возращает Область выходной таблицы.

Синтаксис 1:

```
Область(<Адрес>)
```

Синтаксис 2:

```
Область(<R1>, <C1>, <R2>, <C2>)
```

Англоязычный синоним:

```
Area
```

Параметры:

<Адрес>	Необязательный параметр. Строковое выражение, задающее имя области или адрес в формате «R1C1:R2C2».
<R1>	Необязательный параметр. Числовое выражение. Номер первой строки области.
<C1>	Необязательный параметр. Числовое выражение. Номер первого столбца области.
<R2>	Необязательный параметр. Числовое выражение. Номер последней строки области.
<C2>	Необязательный параметр. Числовое выражение. Номер последнего столбца области.

Возвращаемое значение:
Значение типа «ОбластьТаблицы».
Описание:
Функция Область возвращает значение области выходной таблицы или таблицы в режиме ввода данных. Если последняя строка и последний столбец отсутствуют, то область задана единственной ячейкой. Если строки или столбцы отсутсвуют, то область задана диапазоном столбцов или строк соответственно. Если метод вызван без параметров, то область задана всей таблицей.
Пример:
```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R1C1:R8C16");
```

ПовторятьПриПечатиСтроки

Задать строки результирующей таблицы, повторяющиеся при печати в начале каждой страницы.
Синтаксис:
ПовторятьПриПечатиСтроки(<НачСтрока>, <КонСтрока>)
Англоязычный синоним:
RepeatRowsWhilePrinting
Параметры:

<НачСтрока>	Числовое выражение — номер первой строки повторения.
<КонСтрока>	Числовое выражение — номер последней строки повторения.

Описание:
Метод ПовторятьПриПечатиСтроки позволяет задать строки результирующей таблицы, повторяющиеся при печати в начале каждой страницы.
Пример:
```
Таб.ПовторятьПриПечатиСтроки(1, Таб.ВысотаСекции("ШапкаОтчета"));
```

ПовторятьПриПечатиСтолбцы

Задать столбцы результирующей таблицы, повторяющиеся при печати на каждой странице слева.
Синтаксис:
ПовторятьПриПечатиСтолбцы(<НачСтолбец>, <КонСтолбец>)
Англоязычный синоним:
RepeatColsWhilePrinting
Параметры:

<НачСтолбец>	Числовое выражение — номер первого столбца повторения.
<КонСтолбец>	Числовое выражение — номер последнего столбца повторения.

Описание:
Метод ПовторятьПриПечатиСтолбцы позволяет задать столбцы результирующей таблицы, повторяющиеся при печати на каждой странице слева.
Пример:
```
Таб.ПовторятьПриПечатиСтолбцы(1, Таб.ШиринаСекции("БоковикОтчета"));
```

Опции

Установить флаги вывода сетки, заголовков, фиксации строк и столбцов, опции печати
Синтаксис:
```
Опции(<ВыводСетки>, <ВыводЗаголовков>, <ФиксСтрок>, <ФиксСтолбцов>,
      <ИмяОпцийПечати>, <ИмяСохрРазмОкна>, <ФлагЧерноБелогоПросмотра>,
      <НаправлениеПерехода>)
```
Англоязычный синоним:
Options
Параметры:

<ВыводСетки>	Необязательный параметр. Числовое выражение — флаг вывода сетки: 1 — показывать, 0 не показывать. Значение по умолчанию — 1.
<ВыводЗаголовков>	Необязательный параметр. Числовое выражение — флаг вывода заголовков строк и столбцов: 1 — показывать, 0 не показывать. Значение по умолчанию — 1.
<ФиксСтрок>	Необязательный параметр. Числовое выражение — количество фиксируемых строк. Значение по умолчанию — 0.
<ФиксСтолбцов>	Необязательный параметр. Числовое выражение — количество фиксируемых столбцов. Значение по умолчанию — 0.
<ИмяОпцийПечати>	Необязательный параметр. Строковое выражение — идентификатор набора опций печати для данной таблицы. Значение по умолчанию — пустая строка — в этом случае используются системные опции печати по умолчанию.
<ИмяСохрРазмОкна>	Необязательный параметр. Строковое выражение — идентификатор сохраняемых параметров размеров окна. Если этот параметр указан, то система будет сохранять размер окна и использовать его при следующем открытии табличного документа. Фактически, указывая этот параметр можно указать системе режим сохранения и восстановления размеров окна для данного табличного документа, который иденти-

фицируется этой строкой. Параметр необязателен. Значение по умолчанию — пустая строка — в этом случае размеры окна не запоминаются.

<ФлагЧерноБелого-Просмотра>	Необязательный параметр. Число: 1 — чернобелый просмотр; 0 — обычный режим просмотра. Значение по умолчанию — 0.
<НаправлениеПерехода>	Необязательный параметр. Число: 1 — по строкам, т. е. при вводе данных в ячейки при нажатии клавиши <Enter> будет автоматически выполняться переход к следующей вводимой ячейке в этой строке, а если таковых нет, то в самой левой вводимой ячейке следующей строки; 2 — по столбцам, т. е. при вводе данных в ячейки при нажатии клавиши <Enter> будет автоматически выполняться переход к следующей вводимой ячейке в этом столбце, а если таковых нет, то в самой верхней вводимой ячейке следующего столбца; 3 — при вводе данных в ячейки автоматический переход при нажатии клавиши <Enter> выполняться не будет. Значение по умолчанию 1.

Описание:

Метод Опции позволяет перед показом таблицы установить флаги вывода сетки и вывода заголовков строк и столбцов, а также фиксацию строк и столбцов. Если перед отображением таблицы не использовать метод Опции, то для табличных документов устанавливается режим «только для чтения», а все флаги имеют значение 0.

Даный метод позволяет назначить для каждой таблицы свой собственный набор опций печати, который сохраняется в системе под именем <ИмяОпцийПечати>. Набор опций печати привязан к конкретному рабочему месту (компьютеру) и запоминается всякий раз, когда в режиме исполнения при открытом окне редактирования таблицы пользователь перенастраивает параметры настройки печати (главное меню — «Файл» — «Печать»).

Пример: Если при формировании отчетов в методе Опции используются уникальные имена для хранения настройки печати, то, например, для отчета «Прайс-лист» можно установить в параметрах печати ориентацию бумаги — «портрет», а для отчета «Оборотно-сальдовая ведомость» — «ландшафт». В дальнейшем, при формировании этих отчетов параметры печати будут настраиваться автоматически.

Пример:

```
//Вызов выходного отчета в окно просмотра и редактирования.
Таб.ТолькоПросмотр(1);
Таб.Опции(0, 0, 5, 0, "ОстаткиТоваров");
Таб.Показать("Остатки товаров на складах", "");
```

ОбластьПечати

Устанавливает из языка область печати табличного документа.

Синтаксис:

```
ОбластьПечати(<Верх>, <Лево>, <Низ>, <Право>)
```

Англоязычный синоним:

```
PrintRange
```

Параметры:

<Верх>	Числовое выражение — номер верхней строки таблицы, выводимой на печать.
<Лево>	Числовое выражение — номер крайнего левого столбца, выводимого на печать.
<Низ>	Числовое выражение — номер нижней строки таблицы, выводимой на печать.
<Право>	Числовое выражение — номер крайнего правого столбца, выводимого на печать.

Описание:

Метод ОбластьПечати устанавливает из языка область печати табличного документа

Пример:

```
Таб.Напечатать(0);
```

ПараметрыСтраницы

Установить параметры страницы.

Синтаксис:

```
ПараметрыСтраницы(<Ориентация>, <Масштаб>, <РежимПечатиКопий>, <ПолеСлева>,
                  <ПолеСправа>, <ПолеСверху>, <ПолеСнизу>, <КолонтитулСверху>,
                  <КолонтитулСнизу>, <Автомасштаб> , <ФлагЧерноБелойПечати>,
                  <ИмяПринтера>)
```

Англоязычный синоним:

```
PageSetup
```

Параметры:

<Ориентация>	Необязательный параметр. Числовое выражение — ориентация вывода на печать: 1 — портрет; 2 — ландшафт.
<Масштаб>	Необязательный параметр. Числовое выражение — масштаб (в процентах) вывода на печать.
<РежимПечатиКопий>	Необязательный параметр. Числовое выражение — режим вывода нескольких копий на печать: 0 — (collate) выводить сначала первые страницы всех копий, затем вторые и т. д.;1 — (разобрать) выводить страницы копий по порядку.
<ПолеСлева>	Необязательный параметр. Числовое выражение — расстояние (в миллиметрах)

	от правого края страницы.
<ПолеСверху>	Необязательный параметр. Числовое выражение — расстояние (в миллиметрах) от верхнего края страницы.
<ПолеСнизу>	Необязательный параметр. Числовое выражение — расстояние (в миллиметрах) от нижнего края страницы.
<КолонтитулСверху>	Необязательный параметр. Числовое выражение — размер (в миллиметрах) верхнего колонтитула.
<КолонтитулСнизу>	Необязательный параметр. Числовое выражение — размер (в миллиметрах) нижнего колонтитула.
<Автомасштаб>	Необязательный параметр. Числовое выражение — указывает режим автоматического подбора масштаба для размещения документа при печате на листе по ширине. 1 — Включить; 0 — Выключить. По молчанию — 0.
<ФлагЧерноБелойПечати>	Необязательный параметр. Число: 1 — черно-белая печать; 0 — обычный режим печати. Значение по умолчанию — 0.
<ИмяПринтера>	Необязательный параметр. Строка как в стандартном диалоге печати.

Описание:

Метод ПараметрыСтраницы позволяет установить параметры страницы вывода на печать.

Пример:

```
Таб.ПараметрыСтраницы(1, 100, 1, 17, 17, 25, 25, 10, 10);
```

См. также: Опции

КоличествоЭкземпляров

Установить количество печатаемых экземпляров.

Синтаксис:

```
КоличествоЭкземпляров(<Колич>)
```

Англоязычный синоним:

```
NumberOfCopies
```

Параметры:

<Колич>	Необязательный параметр. Числовое выражение — количество печатаемых экземпляров.

Возвращаемое значение:

Текущее заданное количество печатаемых экземпляров (на момент до исполнения метода).

Описание:

Метод КоличествоЭкземпляров позволяет установить количество печатаемых экземпляров. Если для текущей таблицы используется набор опций печати (см. Опции), то данный метод следует вызывать только после вызова метода Показать, т. к. иначе будут действовать автоматически устанавливаемые настройки печати.

Пример:

```
Таб.КоличествоЭкземпляров(5);
```

См. также: Опции

ЭкземпляровНаСтранице

Установить количество печатаемых экземпляров на странице.

Синтаксис:

```
ЭкземпляровНаСтранице(<Колич>)
```

Англоязычный синоним:

```
CopyesPerPage
```

Параметры:

<Колич>	Необязательный параметр. Числовое выражение — количество печатаемых экземпляров на странице. Может принимать значения: 1 — один экземпляр на странице; 2 — два экземпляра на странице; 0 — автоматический режим размещения двух экземпляров на странице исходя из размеров документа.

Возвращаемое значение:

Текущее заданное количество печатаемых экземпляров на странице (на момент до исполнения метода).

Описание:

Метод ЭкземпляровНаСтранице позволяет установить количество печатаемых экземпляров на странице.

Пример:

```
Таб.ЭкземпляровНаСтранице(2);
```

См. также: Опции

Напечатать

Напечатать таблицу без предварительного просмотра.

Синтаксис:

```
Напечатать(<Флаг>)
```

Англоязычный синоним:

```
Print
```

Параметры:

<Флаг>	Числовое выражение — режим запроса диалога печати: 1 — запрашивать диалог печати; 0 — не запрашивать диалог печати.

Описание:

Метод Напечатать позволяет вывести таблицу на печать без открытия окна редактирования (см. Показать).

Пример:

```
Таб.Напечатать(0);
```

См. также: Показать

Атрибуты и методы объекта «СекцияТаблицы»

<ИмяОбласти>

Возвращает или задает объект «область».

Синтаксис:

<ИмяОбласти>	Идентификатор области секции, как она задана в конфигураторе.

Описание:

В процессе конфигурирования для секции можно задавать практически неограниченное число дополнительно выделенных и поименованных областей для выделения любой необходимой информации.

Атрибут <ИмяОбласти> позволяет обращаться к поименованной области секции. Для обращения к конкретной области секции следует указать ее идентификатор, заданный для этой области в конфигураторе.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
Секция = Таб.ПолучитьСекцию("ЗаголовокОтчета");
Секция.Заглавие.Текст = "Товарный отчет";
Таб.ВывестиСекцию(Секция);
```

Область

Возращает Область секции.

Синтаксис 1:

```
Область(<Адрес>)
```

Синтаксис 2:

```
Область(<R1>, <C1>, <R2>, <C2 >)
```

Англоязычный синоним:

```
Area
```

Параметры:

<Адрес>	Необязательный параметр. Строковое выражение, задающее либо имя области, заданное в конфигураторе, либо адрес в формате «R1C1:R2C2».
<R1>	Необязательный параметр. Числовое выражение. Номер первой строки области.
<C1>	Необязательный параметр. Числовое выражение. Номер первого столбца области.
<R2>	Необязательный параметр. Числовое выражение. Номер последней строки области.
<C2>	Необязательный параметр. Числовое выражение. Номер последнего столбца области.

Возвращаемое значение:

Значение типа «ОбластьТаблицы».

Описание:

Функция Область возвращает значение области секции. При задании адреса области, номера строк и столбцов задаются относительно начала данной секции. Если последняя строка и последний столбец отсутствуют, то область задана единственной ячейкой. Если строки или столбцы отсутсвуют, то область задана диапазоном столбцов или строк соответственно. Если метод вызван без параметров, то область задана всей секцией.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
Секция = Таб.ПолучитьСекцию("ЗаголовокОтчета");
ВыбОбласть = Секция.Область("Заглавие");
ВыбОбласть.Текст = "Товарный отчет";
Таб.ВывестиСекцию(Секция);
```

Атрибуты и методы объекта «ОбластьТаблицы»

Текст

Прочитать/установить значение текста области.

Синтаксис:

```
Текст
```

Англоязычный синоним:

```
Text
```

Описание:

Атрибут Текст позволяет прочитать\установить значение текста области (аналогично тому, как в конфигураторе интерактивно задают значение формулы в свойствах ячейки талины «Свойства»- закладка «Текст»).

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R1C1");
ВыбОбласть.Текст = "Товарный отчет";
```

Расшифровка

Установить расшифровку области.

Синтаксис:

```
Расшифровка(<Значение>, <ФлагДляВсейСтроки>)
```

Англоязычный синоним:

```
Details
```

Параметры:

<Значение>	Значение расшифровки области.
<ФлагДляВсейСтроки>	Необязательный параметр. Число: 0 — обычный режим; 1 — установить данную расшифровку для всей строки; 2 — не вызывать расшифровку для данной ячейки. Значение по умолчанию — 0.

Описание:

Метод Расшифровка записывает расшифровку области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Расшифровка(ВыбДокумент, 1);
```

Объединить

Объединить ячейки области.

Синтаксис:

```
Объединить()
```

Англоязычный синоним:

```
Merge
```

Описание:

Метод Объединить объединяет ячейки области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R1C1:R8C4");
ВыбОбласть.Объединить();
```

Шрифт

Получить/установить шрифт области.

Синтаксис:

```
Шрифт(<ИмяШрифта>)
```

Англоязычный синоним:

```
Font
```

Параметры:

<ИмяШрифта>	Необязательный параметр. Строковое выражение, задающее имя шрифта или «имя, набор символов», где основные наборы символов: 204 — русский; 238 — европейский; 186 — балтийский; 161 — греческий; 162 — турецкий. Если параметр опущен, то шрифт области не изменяется.

Возвращаемое значение:

Имя шрифта до исполнения метода.

Описание:

Метод Шрифт устанавливает шрифт области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial, 204");
```

РазмерШрифта

Получить/установить размер шрифта области.

Синтаксис:

```
РазмерШрифта(<Размер>)
```

Англоязычный синоним:

FontSize

Параметры:

<Размер> Необязательный параметр. Числовое выражение, задающее размер шрифта. Если параметр опущен, то размер шрифта области не изменяется.

Возвращаемое значение:

Размер шрифта до исполнения метода.

Описание:

Метод РазмерШрифта устанавливает размер шрифта области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
```

Полужирный

Получить/установить признак жирного шрифта области.

Синтаксис:

Полужирный(<Жирный>)

Англоязычный синоним:

Bold

Параметры:

<Жирный> Необязательный параметр. Число: 1 — жирный шрифт; 0 — не жирный шрифт. Если параметр опущен, то жирность шрифта области не изменяется.

Возвращаемое значение:

Признак жирного шрифта до исполнения метода.

Описание:

Метод Полужирный устанавливает признак жирного шрифта области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Полужирный(1);
```

Курсив

Получить/установить признак шрифта курсив области.

Синтаксис:

Курсив(<Курсив>)

Англоязычный синоним:

Italic

Параметры:

<Курсив> Необязательный параметр. Число: 1 — шрифт курсив; 0 — шрифт не курсив. Если параметр опущен, то признак курсив шрифта области не изменяется.

Возвращаемое значение:

Признак шрифта курсив до исполнения метода.

Описание:

Метод Курсив устанавливает признак шрифта курсив области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl__1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Курсив(1);
```

Подчеркнутый

Получить/установить признак подчеркнутого шрифта области.

Синтаксис:

Подчеркнутый(<Подчеркнутый>)

Англоязычный синоним:

Underline

Параметры:

<Подчеркнутый> Необязательный параметр. Число: 1 — шрифт подчеркнутый; 0 — шрифт не подчеркнутый. Если параметр опущен, то признак подчеркнутого шрифта области не изменяется.

Возвращаемое значение:

Признак подчеркнутого шрифта до исполнения метода.

Описание:

Метод Подчеркнутый устанавливает признак подчеркнутого шрифта области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
```

ВертикальноеПоложение

Получить/установить признак вертикального выравнивания текста области.

Синтаксис:

ВертикальноеПоложение(<Положение>)

Англоязычный синоним:

VerticalAlign

Параметры:

<Положение>	Необязательный параметр. Число: 1 — сверху; 2 — снизу; 3 — по центру. Если параметр опущен, то признак вертикального выравнивания текста области не изменяется.

Возвращаемое значение:

Признак вертикального выравнивания текста до исполнения метода.

Описание:

Метод ВертикальноеПоложение устанавливает признак вертикального выравнивания текста области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВкбОбласть.Подчеркнутый(1);
ВыбОбласть.ВертикальноеПоложение(3);
```

ГоризонтальноеПоложение

Получить/установить признак горизонтального выравнивания текста области.

Синтаксис:

ГоризонтальноеПоложение(<Положение>)

Англоязычный синоним:

HorizontalAlign

Параметры:

<Положение>	Необязательный параметр. Число: 1 — слева; 2 — справа; 3 — по центру; 4 — по ширине. (5 – 8) — по выделенным столбцам: слева/справа/по центру/по ширине. Если параметр опущен, то признак горизонтального выравнивания текста области не изменяется.

Возвращаемое значение:

Признак горизонтального выравнивания текста до исполнения метода.

Описание:

Метод ГоризонтальноеПоложение устанавливает признак горизонтального выравнивания текста области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть(" tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
```

Контроль

Получить/установить признак контроля текста области.

Синтаксис:

Контроль(<Контроль>)

Англоязычный синоним:

Control

Параметры:

<Контроль>	Необязательный параметр. Число: 1 — Авто; 2 — Обрезать; 3 — Забивать; 4 — Переносить; 5 — Красный; 6 — Забивать+Красный. Если параметр опущен, то признак контроля текста области не изменяется.

Возвращаемое значение:

Признак контроля текста до исполнения метода.

Описание:

Метод Контроль устанавливает признак контроля текста области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
```

РамкаСверху

Получить/установить рамку сверху области.

Синтаксис:

РамкаСверху(<Рамка>)

Англоязычный синоним:

TopBorder

Параметры:

<Рамка> Необязательный параметр. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если параметр опущен, то рамка сверху области не изменяется.

Возвращаемое значение:

Рамка сверху до исполнения метода.

Описание:

Метод РамкаС верху устанавливает рамку сверху области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.РамкаСверху(3);
```

РамкаСнизу

Получить/установить рамку снизу области.

Синтаксис:

РамкаСнизу(<Рамка>)

Англоязычный синоним:

BottomBorder

Параметры:

<Рамка> Необязательный параметр. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если параметр опущен, то рамка снизу области не изменяется.

Возвращаемое значение:

Рамка снизу до исполнения метода.

Описание:

Метод РамкаСнизу устанавливает рамку снизу области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.РамкаСнизу(3 );
```

РамкаСлева

Получить/установить рамку слева области.

Синтаксис:
РамкаСлева(<Рамка>)
Англоязычный синоним:
LeftBorder
Параметры:

<Рамка>	Необязательный параметр. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если параметр опущен, то рамка слева области не изменяется.

Возвращаемое значение:
Рамка слева до исполнения метода.
Описание:
Метод РамкаСлева устанавливает рамку слева области.
Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.РамкаСлева(3);
```

РамкаСправа

Получить/установить рамку справа области.
Синтаксис:
РамкаСправа(<Рамка>)
Англоязычный синоним:
RightBorder
Параметры:

<Рамка>	Необязательный параметр. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если параметр опущен, то рамка справа области не изменяется.

Возвращаемое значение:
Рамка справа до исполнения метода.
Описание:
Метод РамкаСправа устанавливает рамку справа области.
Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.РамкаСправа(3);
```

Рамка

Установить рамки всех ячеек области.
Синтаксис:
Рамка(<РамкаСлева>, <РамкаСверху>, <РамкаСправа>, <РамкаСнизу>)
Англоязычный синоним:
Border
Параметры:

<РамкаСлева> <РамкаСверху> <РамкаСправа> <РамкаСнизу>	Необязательные параметры. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если какой либо параметр опущен, то соответствующая рамка ячеек области не изменяется.

Описание:
Метод Рамка устанавливает рамки всех ячеек области.
Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
```

```
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10 );
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.Рамка(0, 3, 3, 3);
```

РамкаОбвести

Установить рамки по краям области.

Синтаксис:

`РамкаОбвести(<Рамка Слева>, <РамкаСверху>, <РамкаСправа>, <РамкаСнизу>)`

дглоязычный синоним:

`BorderOutline`

Параметры:

`<РамкаСлева>` `<РамкаСверху>` `<РамкаСправа>` `<РамкаСнизу>`	Необязательные параметры. Число: 0 — нет; 1 — очень тонкая; 2 — тонкая точечная; 3 — тонкая сплошная; 4 — средняя сплошная; 5 — толстая сплошная; 6 — двойная; 7 — тонкая средний пунктир; 8 — тонкая длинный пунктир; 9 — толстая пунктир. Если какой либо параметр опущен, то соответствующая рамка ячеек области не изменяется.

Описание:

Метод `РамкаОбвести` устанавливает рамки по краям области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.РамкаОбвести(0, 3, 3, 3);
```

ЦветФона

Получить/установить режим отображения цвета фона области.

Синтаксис 1:

`ЦзетФона(<Цвет>)`

Синтаксис 2:

`ЦветФона(<R>, <G>, <B>)`

Англоязычный синоним:

`BackgroundColor`

Параметры:

`<Цвет>`	Необязательный параметр. Числовое выражение, значение которого задает RGB цвет. Допустимые значения от 0 до 16777'215. Значение -1 (минус единица) задает цвет, заданный для всей области по умолчанию.
`<R>`	Числовое выражение, значение которого задает красную компоненту цвета. Допустимые значения от 0 до 256.
`<G>`	Числовое выражение, значение которого задает зеленую компоненту цвета. Допустимые значения от 0 до 256.
`<B>`	Числовое выражение, значение которого задает синюю компоненту цвета. Допустимые значения от 0 до 256.

Возвращаемое значение:

Текущее числовое значение RGB-цвета фона области (на момент до исполнения метода).

Описание:

Метод `ЦветФона` позволяет установить режим отображения цвета фона области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ЦветФона(34, 126, 211);
```

ЦветРамки

Получить/установить режим отображения цвета рамки области.

Синтаксис 1:

ЦветРамки(<Цвет>)

Синтаксис 2:

ЦветРамки(<R>, <G>, <B>)

Англоязычный синоним:

BorderColor

Параметры:

<Цвет>	Необязательный параметр. Числовое выражение, значение которого задает RGB цвет. Допустимые значения от 0 до 16777'215. Значение -1 (минус единица) задает цвет, заданный для всей области по умолчанию.
<R>	Числовое выражение, значение которого задает красную компоненту цвета. Допустимые значения от 0 до 256.
<G>	Числовое выражение, значение которого задает зеленую компоненту цвета. Допустимые значения от 0 до 256.
<B>	Числовое выражение, значение которого задает синюю компоненту цвета. Допустимые значения от 0 до 256.

Возвращаемое значение:

Текущее числовое значение RGB-цвета рамки области (на момент до исполнения метода).

Описание:

Метод ЦветРамки позволяет установить режим отображения цвета рамки области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ЦветРамки(34, 126, 211);
```

ЦветТекста

Получить/установить режим отображения цвета текста области.

Синтаксис 1:

ЦветТекста(<Цвет>}

Синтаксис 2:

ЦветТекста(<R>, <G>, <B>)

Англоязычный синоним:

TextColor

Параметры:

<Цвет>	Необязательный параметр. Числовое выражение, значение которого задает RGB цвет. Допустимые значения от 0 до 16777'215. Значение -1 (минус единица) задает цвет, заданный для всей области по умолчанию.
<R>	Числовое выражение, значение которого задает красную компоненту цвета. Допустимые значения от 0 до 256.
<G>	Числовое выражение, значение которого задает зеленую компоненту цвета. Допустимые значения от 0 до 256.
<B>	Числовое выражение, значение которого задает синюю компоненту цвета. Допустимые значения от 0 до 256.

Возвращаемое значение:

Текущее числовое значение RGB-цвета текста области (на момент до исполнения метода).

Описание:

Метод ЦветТекста позволяет установить режим отображения цвета текста области.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ЦветТекста(34, 126, 211);
```

ВысотаСтроки

Получить/установить высоту строк, проходящих через область.

Синтаксис:

ВысотаСтроки(<Высота>)

Англоязычный синоним:

RowHeight

Параметры:

<Высота> Необязательный параметр. Число: высота строк, проходящих через область в пунктах с точностью до 0,25. Если параметр опущен, то высота строк области не изменяется.

Возвращаемое значение:

Высота строк области в пунктах области (на момент до исполнения метода).

Описание:

Метод ВысотаСтроки устанавливает высоту строк, проходящих через область.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ВысотаСтроки(3 );
```

ШиринаСтолбца

Получить/установить ширину столбцов, проходящих через область.

Синтаксис:

ШиринаСтолбца(<Ширина>)

Англоязычный синоним:

ColumnWidth

Параметры:

<Ширина> Необязательный параметр. Число: ширина столбцов, проходящих через область в стандартных символах с точностью до 0, 125. Если параметр опущен, то ширина столбцов области не изменяется.

Возвращаемое значение:

Ширина столбцов области в стандартных символах (на момент до исполнения метода).

Описание:

Метод ШиринаСтолбца устанавливает ширину столбцов, проходящих через область.

Пример:

```
Таб = СоздатьОбъект("Таблица");
Таб.Открыть("tabl_1.mxl");
ВыбОбласть = Таб.Область("R8C4");
ВыбОбласть.Шрифт("Arial");
ВыбОбласть.РазмерШрифта(10);
ВыбОбласть.Подчеркнутый(1);
ВыбОбласть.ГоризонтальноеПоложение(3);
ВыбОбласть.Контроль(4);
ВыбОбласть.ШиринаСтолбца(13);
```

Атрибуты и методы таблицы в режиме ввода данных

При работе с таблицей в режиме ввода данных не могут использоваться методы и атрибуты, предназначенные для заполнения таблицы в обычном режиме, например, методы вывода секций, метод Показать и другие, не имеющие смысла для табличного документа в режиме ввода данных. Разумеется, могут использоваться методы управляющие параметрами печати табличного документа, а также собственно метод Напечатать.

В данном разделе приводятся специфические атрибуты и методы, используемые для работы с табличным документом в режиме ввода данных.

<ИмяОбласти>

Предоставляет доступ к именованной области таблицы в режиме ввода данных.

Синтаксис:

<ИмяОбласти> Имя области таблицы в режиме ввода данных.

Описание:

В процессе конфигурирования для таблицы в режиме ввода данных можно задавать практически неограниченное число поименованных областей.

Атрибут <имяОбласти> позволяет обращаться к поименованной области таблицы. Для обращения к конкретной области следует указать ее имя, заданное для этой области в конфигураторе.

Пример:

```
Таблица.Заглавие.Текст = "Товарный отчет";
```

АктивнаяОбласть

Активизировать область таблицы.

Синтаксис:
АктивнаяОбласть(<Адрес>)
Англоязычный синоним:
ActiveArea
Параметры:

<Адрес>	Необязательный параметр. Строковое выражение, задающее имя области или адрес в формате «R1C1:R2C2».

Возвращаемое значение:
Строковое значение имени активной области или адрес в формате «R1C1:R2C2» до начала выполнения метода.
Описание:
Функция АктивнаяОбласть активизирует область таблицы.
Пример:
```
Таблица.АктивнаяОбласть("tabl_1");
```

Выгрузить

Выгружает значения и имена для всех предназначенных для ввода ячеек в объект типа «Список значений».
Синтаксис:
Выгрузить(<СписокЗначений>)
Англоязычный синоним:
Load
Параметры:

<СписокЗначений>	Идентификатор объекта типа «Список значений», куда выгружаются парами значения и имена для всех предназначенных для ввода ячеек.

Описание:
Метод Выгрузить выгружает значения и имена для всех предназначенных для ввода ячеек таблицы в объект типа «Список значений», причем имя ячейки становится представлением а данные из ячейки — значением в записи списка значений.
Пример:
```
Список = СоздатьОбъект("СписокЗначений");
Таблица.Выгрузить(Список);
```

Загрузить

Загружает значения и имена для всех предназначенных для ввода ячеек из объекта типа «Список значений».
Синтаксис:
Загрузить(<СписокЗначений>)
Англоязычный синоним:
Save
Параметры:
<СписокЗначений> Идентификатор объекта типа «Список значений», откуда загружаются парами значения и имена для всех предназначенных для ввода ячеек.
Описание:
Метод Загрузить загружает значения и имена для всех предназначенных для ввода ячеек таблицы из объекта типа «Список значений», причем представление списка значений становится именем ячейки, а значение записи становится значением ячейки.
Пример:
```
Процедура ОчиститьТаблицу()
   Список = СоздатьОбъект("СписокЗначений");
   Таблица.Выгрузить(Список);
   ИмяЯчейки = "";
   Для Н = 1 По Список.РазмерСписка() Цикл
      ЗначЯчейки = Список.ПолучитьЗначение(Н, ИмяЯчейки);
      Список.УстановитьЗначение(Н, 0, ИмяЯчейки, 1);
   КонецЦикла;
   Таблица.Загрузить(Список);
КонецПроцедуры;
```

Атрибуты и методы области таблицы в режиме ввода данных

При работе с областью таблицы в режиме ввода данных не может использоваться метод Расшифровка, а также метод Объединить.

В данном разделе приводятся специфические атрибуты и методы, используемые для работы с областью таблицы в режиме ввода данных.

Значение

Арибут, предоставляет доступ к значению, записанному в области.
Синтаксис:
Значение

Англоязычный синоним:
Value
Описание:
Атрибут Значение позволяет прочитать\установить значение, записанное в области (аналогично тому, как в конфигураторе интерактивно задают значение в свойствах ячейки талины «Свойства»- закладка «Данные»).
Пример:
```
ВыбОбласть = Таблица.Область("R1C1");
ВыбОбласть.Значение = '01.09.98';
```

Формат

Устанавливает формат для вывода всех выражений в ячейках области таблицы.
Синтаксис:
Формат(<СтрокаФормата>)
Англоязычный синоним:
Format
Параметры:

<СтрокаФормата>	Необязательный параметр. Строковое выражение, содержащее форматную строку (см. Формат).

Возвращаемое значение:
Строковое значение, содержащее текущую форматную строку по умолчанию для области таблицы (на момент до исполнения метода).
Описание:
Метод Формат устанавливает формат для вывода всех выражений выводимых в ячейках области таблицы.
Пример:
```
ВыбОбласть.Формат("415.2");
```

УстановитьТип

Установить тип для значения ячейки неопределенного вида.
Синтаксис:
УстановитьТип(<Выражение>)
Англоязычный синоним:
AssignType
Параметры:

<Выражение>	Выражение. Тип значения этого выражения будет присвоен значению ячейки.

Описание:
Метод УстановитьТип позволяет установить тип для значения ячейки, которому в конфигураторе назначен тип «Неопределенный».
Пример:
```
ВыбОбласть.УстановитьТип(Товар);
```
См. также: НазначитьТип, ТипЗначения, ТипЗначенияСтр

НазначитьТип

Назначить тип для значения ячейки неопределенного вида.
Синтаксис:
НазначитьТип(<ИмяТипа>, <Длина>, <Точность>)
Англоязычный синоним:
SetType
Параметры:

<ИмяТипа>	Строковое выражение — название типа данных, которое назначается значению ячейки. 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 значения ячейки, которому в конфигураторе назначен тип «Неопределенный».
Пример:
```
ВыбОбласть.НазначитьТип("Число", 15, 2);
```
См. также: УстановитьТип, ТипЗначения, ТипЗначенияСтр

Доступность

Установка режима редактирования.
Синтаксис:
Доступность(<Режим>)

Англоязычный синоним:
Enable
Параметры:
<Режим> Необязательный параметр. Числовое выражение: 1 — разрешено редактирование элемента формы; 0 — запрещено редактирование.
Возвращаемое значение:
Текущее числовое значение режима редактирования области таблицы (на момент до исполнения метода).
Описание:
Метод Доступность позволяет установить режим редактирования выбранной области таблицы.
Пример:
```
ВыбОбласть.Доступность(0);
```

Редактирование

Определяет возможность редактирования значения области таблицы.
Синтаксис:
Редактирование(<Флаг>)
Англоязычный синоним:
EnableEdit
Параметры:
<Флаг> Число: 1 — значения ячеек области таблицы редактируются как обычно; 0 — значения не редактируется но могут выбираться кнопкой выбора. Отличие от метода Доступность в том, что Доступность отключает и кнопку выбора.
Описание:
Метод Редактирование определяет возможность редактирования значения непосредственно в ячейках области таблицы для полей ввода типа «Число», «Строка», «Дата», «Счет».
Пример:
```
ВыбОбласть.Редактирование(1);
```

Системные предопределенные процедуры работы с таблицами

ОбработкаЯчейкиТаблицы

Предопределенная процедура обработки ячейки таблицы.
Описание:
```
ОбработкаЯчейкиТаблицы(<Значение>, <ФлагСтандартнойОбработки> , <КонтекстТаблицы>,
                       <Адрес>)
```
Англоязычный синоним:
SheetCellProcessing
Параметры:

<Значение>	Здесь в процедуру передается вычисленное значение ячейки (задается в конфигураторе: «Свойства» ячейки, «Текст», «Значение»).
<ФлагСтандартнойОбработки>	Флаг, установка которого в теле процедуры в 1 приведет к выполнению стандартной обработки значения ячейки (открытию документа, элемента справочника и т. п.).
<КонтекстТаблицы>	Необязательный параметр. Имя переменной, куда система передаст контекст всей таблицы (см. Передача контекста в качестве параметра). С помощью значения этого контекста можно произвольно манипулировать данной таблицей пока она открыта, например, вывести туда дополнительные секции или очистить и заполнить всю таблицу заново. Пока данный объект «Таблица» существует, тип значения данного параметра равен 100 (см. ТипЗначения), если закрыта — 0.
<Адрес>	Необязательный параметр. Имя переменной, куда система передаст адрес ячейки/объекта в формате «R1C1:R2C2».

Описание:
Вызов процедуры ОбработкаЯчейкиТаблицы на исполнение производится в системе 1С:Предприятие по двойному щелчку мыши или по нажатию клавиши «Enter» в табличном документе на выбранной ячейке

Замечание ОбработкаЯчейкиТаблицы — предопределенная процедура. Она не является встроенной процедурой языка. Для нее определено только название и синтаксис. Тело процедуры должно быть написано пользователем, осуществляющим конфигурирование, в программном модуле формы, из которой сформирован табличный документ или в глобальном модуле.

Замечание Объект типа «СписокЗначений» может записываться в поле «значение» ячейки таблицы и использоваться затем процедурой ОбработкаЯчейкиТаблицы.

Важно! Если процедура описана в модуле формы, то вызывается она, иначе система запускает одноименную процедуру из глобального модуля.

Внимание! Данная предопределенная процедура не вызывается при выборе ячейки таблицы в режиме ввода данных. Для этого случая вызывается предопределенная процедура ПриВыбореЯчейкиТаблицы.

Пример:

```
Процедура ОбработкаЯчейкиТаблицы(ЗначениеЯчейки, Флаг)
    // Тело процедуры
    //. . .
КонецПроцедуры
```

ПриВыбореЯчейкиТаблицы

Предопределенная процедура обработки ячейки таблицы в режиме ввода данных.

Описание:

ПриВыбореЯчейкиТаблицы(<ИмяИлиАдрес>, <Значение>)

Англоязычный синоним:

OnSelectSheetCell

Параметры:

<ИмяИлиАдрес>	Здесь в процедуру системой передается строковое значение имени области таблицы, если выбранная ячейка помечена в таблице как отдельная область, или адрес ячейки в формате «R1C1:R2C2».
<Значение>	Здесь в процедуру системой передается значение даной ячейки, а если диаграмма — значение выбранного элемента.

Описание:

Вызов процедуры ПриВыбореЯчейкиТаблицы на исполнение производится в системе 1С:Предприятие по двойному щелчку мыши или по нажатию клавиши «Enter» на выбранной ячейке в таблице

Внимание! Данная предопределенная процедура вызывается в модуле формы при выборе ячейки только для таблиц в режиме ввода данных.

Замечание ПриВыбореЯчейкиТаблицы — предопределенная процедура. Она не является встроенной процедурой языка. Для нее определено только название и синтаксис. Тело процедуры должно быть написано пользователем, осуществляющим конфигурирование, в программном модуле формы, в которой используется таблица в режиме ввода данных.

Пример:

```
Процедура ПриВыбореЯчейкиТаблицы(ИмяЯчейки, Знач)
    // Тело процедуры
    // ...
КонецПроцедуры
```

Глава 32
Работа с Текстом

Для работы с текстами в системе используется специальный агрегатный тип данных — «Текст». Механизм работы с текстами предназначен для формирования отчетов в текстовом виде, а также для обмена информацией с другими программами посредством текстовых файлов. Средства языка имеют возможность не только вывода строк в текстовые файлы, но и считывания имеющихся файлов с последующим разбором его по строкам.

Контекст работы с текстом

Во всех программных модулях вызов методов текста может выполняться только при помощи переменной со ссылкой на объект типа «Текст». Объект создается при помощи функции `СоздатьОбъект`, ссылка на который присваивается переменной. Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

При создании переменной типа «Текст» функции `СоздатьОбъект` в качестве параметра передается ключевое слово `"Текст"`.

Англоязычный синоним ключевого слова `Текст` — `Text`.

Пример:

```
МойОтчет = СоздатьОбъект("Текст");
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("List.txt");
```

Методы текста

КоличествоСтрок

Возвратить количество строк текста.

Синтаксис:

```
КоличествоСтрок()
```

Англоязычный синоним:

```
LinesCnt
```

Возвращаемое значение:

Количество строк в тексте.

Описание:

Метод `КоличествоСтрок` возвращает количество строк в тексте.

Пример:

```
Процедура ЗагрузкаПродукцииПоКаталогу()
   Прод = СоздатьОбъект("Справочник.ГотоваяПродукция");
   Если ВыбГруппа.Выбран() = 0 Тогда
      Предупреждение("Ошибка! Не выбрана группа!");
      Возврат;
   ИначеЕсли ВыбГруппа.ЭтоГруппа() = 0 Тогда
      Предупреждение("Не выбрана группа! Загрузка в корень!");
   Иначе
      Тов.ИспользоватьРодителя(ВыбГруппа);
   КонецЕсли;
   Текст = СоздатьОбъект("Текст");
   Текст.Открыть(ИмяФайла);
   Если Текст.КоличествоСтрок() = 0 Тогда
      Предупреждение("Текст пустой!");
      Возврат;
   КонецЕсли;
   Для Ном = 1 По Текст.КоличествоСтрок() Цикл
      Стр = Текст.ПолучитьСтроку(Ном);
      Поле = 0;
      Пока СтрДлина(Стр) > 0 Цикл
         Поле = Поле + 1;
         Инд = Найти(Стр, "-");
         Если Инд > 0 Тогда
            Стр1 = Сред(Стр, 2, Инд - 3);
            Стр = Сред(Стр, Инд + 1);
         Иначе
            Стр1 = Сред(Стр, 2, (СтрДлина(Стр) - 2));
            Стр = "";
         КонецЕсли;
         Если Поле = 1 Тогда
            Код = Стр1;
         ИначеЕсли Поле = 2 Тогда
```

```
            Имя = Стр1;
         КонецЕсли;
      КонецЦикла;
      Прод.НоваяГруппа();
      Сообщить(Формат(Ном, "ЧЗ") + "-загрузка: " + Формат(Код, "45") + ":" + Имя);
      Прод.Наименование = Имя;
      Прод.Код = Число(Код);
      Прод.Записать();
      Сообщить(" — Загружен!");
   КонецЦикла;
КонецПроцедуры
```

ПолучитьСтроку

Полнить строку текста по номеру.

Синтаксис:

`ПолучитьСтроку(<НомерСтроки>)`

Англоязычный синоним:

`GetLine`

Параметры:

`<НомерСтроки>`	Числовое выражение — номер строки.

Возвращаемое значение:

Строковое значение — требуемая строка текста.

Описание:

Метод `ПолучитьСтроку` возвращает значение строки с номером `<НомерСтроки>`.

Пример:

```
Процедура Тест()
   Текст = СоздатьОбъект("Текст");
   Текст.Открыть(ИмяФайла);
   Если Текст.КоличествоСтрок() = 0 Тогда
      Предупреждение("Текст пустой!");
      Возврат;
   КонецЕсли;
   Для Ном = 1 по Текст.КоличествоСтрок() Цикл
      Стр = Текст.ПолучитьСтроку(Ном);
      Сообщить(Строка(Ном) + Стр);
   КонецЦикла;
КонецПроцедуры
```

Открыть

Открыть текстовый файл.

Синтаксис:

`Открыть(<ИмяФайла>)`

Англоязычный синоним:

`Open`

Параметры:

`<ИмяФайла>`	Строковое выражение — имя файла.

Описание:

Метод `Открыть` открывает файл с именем `<ИмяФайла>`.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
```

Шаблон

Установить флаг добавления строк по шаблону.

Синтаксис:

`Шаблон(<Флаг>)`

Англоязычный синоним:

`Template`

Параметры:

`<Флаг>`	Необязательный параметр. Числовое выражение: 1 — установить флаг добавления строк по шаблону, 0 — отменить.

Возвращаемое значение:

Текущее числовое значение флага добавления строк по шаблону (на момент до исполнения метода).

Описание:

Метод `Шаблон` устанавливает флаг при котором все добавления строк в текст выполняются с заменой полей ограниченных квадратными скобками на значения содержащихся в них выражений (см. гл. «Системные процедуры и функции», функция `Шаблон`).

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.Шаблон(1);
ТекстДок.ЗаменитьСтроку(52, "[Услуга.Наименование],
                        арт.[Услуга.Код].....[Услуга.Цена]");
ТекстДок.Шаблон(0);
ТекстДок.ВставитьСтроку(53, "Вывоз мусора, 6321, 58.000");
ТекстДок.ТолькоПросмотр(1);
```

См. также: «Системные процедуры и функции», функция Шаблон

ФиксШаблон

Установить флаг добавления строк по фиксированному шаблону.

Синтаксис:

ФиксШаблон(<Флаг>)

Англоязычный синоним:

FixTemplate

Параметры:

<Флаг>	Числовое выражение: 1 — установить флаг добавления строк по фиксированному шаблону, 0 — снять флаг.

Возвращаемое значение:

Текущее числовое значение флага добавления строк по фиксированному шаблону (на момент до исполнения метода).

Описание:

Метод ФиксШаблон устанавливает флаг при котором все добавления строк в текст выполняются с заменой полей ограниченных квадратными скобками на значения содержащихся в них выражений (см. гл. «Системные процедуры и функции», функция ФиксШаблон).

В отличие от метода Шаблон, ограниченные квадратными скобками 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.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.ФиксШаблон(1);
ТекстДок.ЗаменитьСтроку(52, "[Услуга.Наименование],
                         арт.[Услуга.Код].....[Услуга.Цена]");
ТекстДок.Шаблон(0);
ТекстДок.ВставитьСтроку(53, "Вывоз мусора, 6321, 58.000");
ТекстДок.ТолькоПросмотр(1);
```

См. также: «Системные процедуры и функции», функция ФиксШаблон

ВставитьСтроку

Вставить строку с указанным номером.

Синтаксис:

ВставитьСтроку(<НомерСтроки>, <Строка>)

Англоязычный синоним:

InsertLine

Параметры:

<НомерСтроки>	Числовое выражение — номер вставляемой строки.
<Строка>	Строковое выражение.

Описание:

Метод ВставитьСтроку вставляет в текст строку <Строка> с номером <НомерСтроки>.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.ЗаменитьСтроку(52, "Установка оборудоввания 3456....." + Стоимость);
ТекстДок.ВставитьСтроку(53, "Замена оборудования 5691... ");
ТекстДок.ТолькоПросмотр(1);
```

ДобавитьСтроку

Добавить строку в конец текста.

Синтаксис:

ДобавитьСтроку(<Строка>)

Англоязычный синоним:

AddLine

Параметры:

<Строка> Строковое выражение.

Описание:

Метод ДобавитьСтроку добавляет в конец текста строку <Строка>.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.ДобавитьСтроку("Работы ..., 3056 ..75 000 руб.");
ТекстДок.ТолькоПросмотр(1).
```

ЗаменитьСтроку

Заменить строку с указанным номером.

Синтаксис:

```
ЗаменитьСтроку(<НомерСтроки>, <Строка>)
```

Англоязычный синоним:

```
ReplaceLine
```

Параметры:

<НомерСтроки> Числовое выражение — номер замещаемой строки.

<Строка> Строковое выражение.

Описание:

Метод ЗаменитьСтроку замещает строку текста с номером <НомерСтроки> на строку <Строка>.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.ЗаменитьСтроку(52, "Работы ..., 3056 ..75 000 руб.");
ТекстДок.ТолькоПросмотр(1).
```

УдалитьСтроку

Удалить строку с указанным номером.

Синтаксис:

```
УдалитьСтроку(<НомерСтроки>)
```

Англоязычный синоним:

```
DeleteLine
```

Параметры:

<НомерСтроки> Числовое выражение — номер удаляемой строки.

Описание:

Метод УдалитьСтроку удаляет из текста строку с номером <НомерСтроки>.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.УдалитьСтроку(5 3);
```

ТолькоПросмотр

Установить режима редактирования текста.

Синтаксис:

```
ТолькоПросмотр(<Режим>)
```

Англоязычный синоним:

```
Readonly
```

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 1 — запрещено редактирование текста, 0 — разрешено редактирование текста.

Возвращаемое значение:

Текущее числовое значение режима редактирования текста (на момент до исполнения метода).

Описание:

Метод ТолькоПросмотр позволяет установить режим редактирования текста в окне редактирования.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.ТолькоПросмотр(1);
ТекстДок.Показать("Редактирование Прайс-листа", "catalog.txt");
```

Показать

Открыть окно редактирования текста.

Синтаксис:

```
Показать(<Заголовок>, <ИмяФайла>)
```

Англоязычный синоним:

```
Show
```

Параметры:

<Заголовок> Заголовок окна редактирования.
<ИмяФайла> Строковое выражение с именем файла.

Описание:

Метод Показать открывает окно с текстом для редактирования и просмотра. Если параметр <ИмяФайла> имеет непустое значение, то при закрытии окна система будет предлагать сохранить данные в указанный файл. Если файла с именем <ИмяФайла> не существует, то будет создан новый файл с таким именем для сохранения текста. Если параметр <ИмяФайла> опущен или имеет пустое значение, то при закрытии окна система не будет предлагать сохранить данные в файл. Это имеет смысл для текстовых документов, которые формируются только для просмотра или печати, и их не обязательно записывать в файл. Разумеется, после открытия окна пользователь в любом случае может записать такой текст в файл, используя команды главного меню «Файл».

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.Показать("Редактирование Прайс-листа", "catalog.txt");
```

Очистить

Очищает содержимое текстового документа.

Синтаксис:

```
Очистить()
```

Англоязычный синоним:

```
Clear
```

Описание:

Метод Очистить очищает содержимое текстового документа. Его использование позволяет заново заполнить содержимое текстового документа.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.Очистиь();
```

КодоваяСтраница

Установить/определить режим кодировки.

Синтаксис:

```
КодоваяСтраница(<Режим>)
```

Англоязычный синоним:

```
SetCodePage
```

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 0 — Windows-кодировка, 1 — DOS-кодировка. Если параметр не задан, то режим кодировки не меняется (используется для определения текущего режима кодировки без его смены).

Возвращаемое значение:

Текущее числовое значение режима кодировки (на момент до исполнения метода).

Описание:

Метод КодоваяСтраница позволяет установить режим кодировки для чтения и записи строковых значений в файл.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.КодоваяСтраница(0);
ТекстДок.Открыть("catalog.txt");
ТекстДок.Показать("Редактирование Прайс-листа", "catalog.txt");
```

Записать

Записать текст в файл с указанным именем.

Синтаксис:

```
Записать(<ИмяФайла>)
```

Англоязычный синоним:

```
Write
```

Параметры:

<ИмяФайла> Строковое выражение — имя файла.

Описание:

Метод Записать записывает текст в файл с именем <ИмяФайла>.

Пример:

```
ТекстДок = СоздатьОбъект("Текст");
ТекстДок.Открыть("catalog.txt");
ТекстДок.Записать("price.txt");
```

Глава 33
Работа с Запросами

Для формирования и выполнения запросов к документам, справочникам, регистрам, журналам расчетов, планам счетов, бухгалтерским операциям и проводкам в системе используется специальный агрегатный тип данных — «Запрос». Возможности работы со справочниками, документами и журналами расчетов предоставляют достаточно мощные средства получения различной информации об этих объектах. Однако, существует также необходимость в получении информации, сгруппированной определенным образом, которую невозможно или очень сложно получить непосредственно работая с документами, справочниками, регистрами или журналами расчетов. Для получения такой информации и существует механизм запросов.

Одним из классических примеров его применения может служить сводка по состоянию регистра на конкретный момент времени. В более сложных запросах возможно получение сгруппированной информации по справочникам, журналам расчетов и документам.

Кроме того, механизм запросов позволяет легко получать информацию в различных временных разрезах.

Контекст работы с запросами

Во всех программных модулях доступ к атрибутам и методам запросов может выполняться только через переменную, созданную функцией `СоздатьОбъект`. Чтобы вызвать атрибут или метод объекта, имя этого атрибута, метода (с указанием необходимых параметров) пишется через точку после имени переменной.

При создании объекта типа «Запрос» в качестве параметра функции `СоздатьОбъект` используется ключевое слово `Запрос` .

Англоязычный синоним ключевого слова `Запрос` — `Query`.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
```

Структура запросов и методика их использования

Использование запросов позволяет легко строить простые отчеты и облегчает построение сложных отчетов. При построении сложных отчетов использование запросов может существенно снизить трафик сети, т.к. однажды выбранная лри исполнении запроса информация, хранящаяся во временном наборе данных на локальном компьютере, может многократно использоваться.

Запросы можно использовать не только для построения отчетов, но и для реализации других процедур конфигурации, требующих получения из БД некой сводной информации. Например, реализация алгоритма списания стоимости товара по методам FIFO или LIFO.

Работа с запросами предполагает следующий порядок:

- при помощи функции `СоздатьОбъект` создается объект типа «Запрос» и ссылка на него присваивается какой-либо переменной. Далее обращение к запросу производится посредством этой ссылки.
- после создания переменной типа «Запрос» следует обращение к методу `Выполнить`, которому в качестве параметра передается текст запроса, написанный на специальном языке запросов (см. главу «Язык запросов»). Метод `Выполнить` анализирует текст запроса, выполняет в соответствии с ним выборку данных и формирует временный выходной набор данных (выборку).
- после этого организуется циклическая обработка сформированного временного набора данных (выборки) с целью получения требуемого отчета.

В данном разделе мы рассмотрим структуру создаваемого запросом временного набора данных и работу методов запросов по выборке информации из этого временного набора.

Дальнейшее описание проведем на простом примере: необходимо получить отчет о количестве товаров, хранящихся на складах. Текст процедуры, выполняющей эту операцию приведен ниже.

Пример:

```
НашЗапрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|СКЛАД = Регистр.ТоварныйЗапас.Склад;
|ТОВАР = Регистр.ТоварныйЗапас.Товар;
|КОЛИЧЕСТВО = Регистр.ТоварныйЗапас.Количество;
|Группировка ТОВАР Упорядочить По ТОВАР.Код;
|Группировка СКЛАД Упорядочить По СКЛАД.Код;
|Функция КОЛ = КонОст(КОЛИЧЕСТВО);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если НашЗапрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Структура временного набора данных, созданная запросом "`НашЗапрос`" после его выполнения, представлена на следующем рисунке:

Товар	Склад	Кол.

Итог общий	X	X	62
Итог по товару Стол	Стол	X	35
Строка по товару Стол на складе Первый	Стол	Первый	10
Строка по товару Стол на складе Второй	Стол	Второй	20
Строка по товару Стол на складе Третий	Стол	Третий	5
Итог по товару Стул	Стул	X	9
Строка по товару Стул на складе Второй	Стул	Второй	7
Строка по товару Стул на складе Третий	Стул	Третий	2
Итог по товару Шкаф	Шкаф	X	18
Строка по товару Шкаф на складе Первый	Шкаф	Первый	8
Строка по товару Шкаф на складе Третий	Шкаф	Третий	10

После выполнения запроса в программе можно использовать полученный временный набор данных. Изначально объект "НашЗапрос" спозиционирован на первой записи временного набора данных, где содержится общий итог по запросу. Поэтому общие итоги по запросу можно использовать сразу же после выполнения запроса.

Далее, допустим, в цикле мы проходим только по самой внешней группировке запроса: "Товар". В теле этого цикла объект "НашЗапрос" позиционируется во временном наборе данных на записи, содержащие итоги по каждому конкретному товару, поэтому в теле этого цикла мы можем использовать итоги по товарам.

После завершения цикла по самой внешней группировке, объект "НашЗапрос" снова позиционируется на первой записи временного набора данных, где содержится общий итог по запросу. Поэтому, общие итоги по запросу можно использовать в двух местах: до и после цикла по самой внешней группировке запроса.

Продолжение примера:

```
//Итог общий
Пока НашЗапрос.Группировка("Товар") = 1 Цикл
   //Итог по товару
КонецЦикла;
//Итог общий
```

Поскольку после первого просмотра временного набора данных (как мы это сделали выше) объект "НашЗапрос" снова спозиционирован на первой записи, то можно запустить просмотр еще раз.

Допустим, теперь нам надо просмотреть в цикле всю информацию по внешней ("Товар") и вложенной группировке запроса ("Склад"). В теле цикла по внешней группировке ("Товар") объект "НашЗапрос" позиционируется во временном наборе данных на записи, содержащие итоги по каждому конкретному товару, поэтому в теле этого цикла мы можем использовать итоги по товарам.

В теле цикла по вложенной группировке ("Склад") объект "НашЗапрос" позиционируется во временном наборе данных на записи, содержащие строки по каждому конкретному товару на конкретном складе, поэтому в теле этого цикла мы можем использовать данные о количестве товара на складе.

После завершения цикла по вложенной группировке ("Склад"), объект "НашЗапрос" снова позиционируется на записи временного набора данных, где содержится общий итог по конкретному товару. Поэтому, общие итоги по конкретному товару можно использовать в двух местах: до и после цикла по вложенной группировке ("Склад").

После завершения цикла по самой внешней группировке, объект "НашЗапрос" снова позиционируется на первой записи временного набора данных, где содержится общий итог по запросу. Поэтому, общие итоги по запросу можно использовать в двух местах: до и после цикла по самой внешней группировке запроса.

Продолжение примера:

```
//Итог общий
Пока НашЗапрос.Группировка("Товар") = 1 Цикл
   //Итог по товару
   Пока НашЗапрос.Группировка("Склад") = 1 Цикл
      //Строка по товару — складу
   КонецЦикла;
   //Итог по товару
КонецЦикла;
//Итог общий
```

После выхода из процедуры, где была определена переменная, содержащая объект типа «Запрос» (в нашем примере "НашЗапрос") или после уничтожения объекта типа «Запрос» (НашЗапрос = 0;), временный набор данных на локальной машине пользователя уничтожается.

Из приведенного выше примера можно сделать несколько важных заключений:

- при просмотре временного набора данных, вложенность циклов просмотра должна точно повторять порядок группировок запроса (нельзя входить во вложенную группировку, не войдя в предыдущую);
- допускается просматривать временный набор данных, опуская с некоторого уровня все вложенные (внутренние) группировки;
- не следует прерывать последовательность просмотра временного набора данных (например, оператором Прервать;), если вы собираетесь использовать временный набор дальше или еще раз, т. к. в таком случае теряется точка позиционирования во временном наборе и продолжать просмотр невозможно;

Пример:

```
Пока Запрос.Группировка("Товар") = 1 Цикл
   Если (Запрос.Товар = НужныйТовар) Тогда
```

```
        Пока Запрос.Группировка("Отдел") = 1 Цикл
           Если (Запрос.Отдел = НужныйОтдел) Тогда
              Пока Запрос.Группировка("Сотрудник") = 1 Цикл
                 Если (Запрос.Сотрудник = НужныйСотрудник) Тогда
                    Пока Запрос.Группировка("Док") = 1 Цикл
                       Док = Запрос.Док;
                       Если Запрос.ПрихКол <> 0 Тогда
                          Таб.ВывестиСекцию("Приход");
                       ИначеЕсли Запрос.РасхКол <> 0 Тогда
                          Приращение = Запрос.РасхКол;
                          Таб.ВывестиСекцию("Расход");
                       КонецЕсли;
                    КонецЦикла;
                 КонецЕсли;
              КонецЦикла;
           КонецЕсли;
        КонецЦикла;
     КонецЕсли;
  КонецЦикла;
```

Атрибуты запросов

Атрибутами запроса являются объявленные в описании запроса внутренние переменные (см. главу «Язык запросов»), имена группировок и функций запроса. Все атрибуты запросов — только для чтения.

Чтобы обратиться к атрибуту запроса, имя этого атрибута пишется через точку после имени ссылки на запрос. Значения атрибутов запроса определяются текущим положением в полученной выборке.

Методы запросов

Выполнить

Выполнить запрос.

Синтаксис:

`Выполнить(<ТекстЗапроса>)`

Англоязычный синоним:

`Execute`

Параметры:

`<ТекстЗапроса>`	Строковое выражение, содержащее текст запроса на языке запросов (см. Главу «Язык запросов»).

Возвращаемое значение:

Число: 1 — если запрос выполнен успешно, 0 — если зафиксирована ошибка при выполнении запроса (синтаксическая или времени выполнения).

Описание:

Метод `Выполнить` анализирует описание запроса, содержащееся в тексте запроса `<ТекстЗапроса>`, выполняет выборку данных, формирует временный выходной набор данных (выборку), вычисляет значения функций запроса (см. главу «Язык запросов»).

Язык запросов предназначен для описания запросов к базе данных. Написанный на языке запросов текст описания передается методу `Выполнить` в качестве параметра. Метод `Выполнить` непосредственно выполняет запрос, о результате его выполнения формируется временный выходной набор данных, который в дальнейшем используется для заполнения формы отчета.

Пример:

```
Процедура Сформировать()
   // сформируем данные на начало месяца
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   _Дата = ЖР.НачалоТекущегоПериода();
   // Создание объектов типа "Запрос"
   Запрос = СоздатьОбъект("Запрос");
   ЗапросКат = СоздатьОбъект("Запрос");
   флаг1 = Запрос.Выполнить(
   "//({ЗАПРОС(Двойной)
   |Период с _Дата по _Дата;
   |Оклад = Справочник.Сотрудники.Оклад;
   |Пдр = Справочник.Сотрудники.МестоРаботы.Владелец;
   |Ктг = Справочник.Сотрудники.Категория;
   |Условие((Ктг.Выбран() = 1) И (Пдр.Выбран() = 1));
   |Группировка Пдр без групп;
   |Группировка Ктг без групп;
   |Функция Всего = Сумма(Оклад);
   |"//}} ЗАПРОС
```

```
    );
    Флаг2 = ЗапросКат.Выполнить(
    "//{{ЗАПРОС(Одинарный)
    |Период с _Дата по _Дата;
    |Оклад = Справочник.Сотрудники.Оклад;
    |Ктг = Справочник.Сотрудники.Категория;
    |Условие(Ктг.Выбран() = 1);
    |Группировка Ктг без групп;
    |Функция Всего = Сумма(Оклад);
    |"//}}ЗАПРОС
    );
    // Если ошибка в запросе, то выход из процедуры
    Если ((Флаг1 = 0) ИЛИ (Флаг2 = 0)) Тогда
       Сообщить("Ошибка в запросе!");
       Возврат;
    КонецЕсли;
    // Подготовка к заполнению выходных форм данными запроса
    Таб = СоздатьОбъект("Таблица");
    // Выводим заглавие таблицы
    Таб.ВывестиСекцию("Документ<|ДокументВерт<");
    Пока ЗапросКат.Группировка("Ктг") = 1 Цикл
       Таб.ПрисоединитьСекцию("Документ<|Осн2");
    КонецЦикла;
    Таб.ПрисоединитьСекцию("Документ<|ДокументВерт>");
    // Выводим колонтитул таблицы
    Таб.ВывестиСекцию("КолонТитул|ДокументВерт<");
    Пока ЗапросКат.Группировка("Ктг") = 1 Цикл
       Таб.ПрисоединитьСекцию("КолонТитул|Осн2");
    КонецЦикла;
    Таб.ПрисоединитьСекцию("КолонТитул|ДокументВерт>");
    Продолжать = 1;
    Пока Запрос.Группировка("Пдр") = 1 Цикл
       // Заполнение полей
       Пдр Таб.ВывестиСекцию("Осн1|ДокументВерт<");
       Далее = 1;
       Пока Продолжать = 1 Цикл
          // Заполнение полей Ктг
          СлКат = ЗапросКат.Группировка("Ктг");
          Если Далее = 1 Тогда
             ОК = Запрос.Группировка("Ктг");
          КонецЕсли;
          Если СлКат = 0 Тогда
             Прервать;
          КонецЕсли;
          Если ЗапросКат.Ктг = Запрос.Ктг Тогда
             Таб.ПрисоединитьСекцию("Осн1|Осн2");
             Далее = 1;
          Иначе
             Таб.ПрисоединитьСекцию("Осн1|Пусто");
             Далее = 0;
          КонецЕсли;
       КонецЦикла;
       Таб.ПрисоединитьСекцию("Осн1|ДокументВерт>");
    КонецЦикла;
    // Заполнение полей "Итого"
    Таб.ВывестиСекцию("Документ>|ДокументВерт<");
    Пока ЗапросКат.Группировка("Ктг") = 1 Цикл
       Таб.ПрисоединитьСекцию("Документ>|Осн2");
    КонецЦикла;
    Таб.ПрисоединитьСекцию("Документ>|ДокументВерт>");
    // Вывод заполненной формы Таб.Опции(0, 0, 0, 0);
    Таб.ТолькоПросмотр(1);
    Таб.Показать("Результат", );
КонецПроцедуры
```

ИспользоватьГрафуОтбора

Установить режим использования графы отбора в запросе.

Синтаксис:

```
ИспользоватьГрафуОтбора(<ГрафаОтбора>)
```

Англоязычный синоним:
`UseSelectionColumn`
Параметры:

`<ГрафаОтбора>`	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.

Возвращаемое значение:
Строковое значение: идентификатор использованной реально графы отбора, если метод вызывается после выполнения запроса.
Описание:
Метод устанавливает режим использования графы отбора в запросе. Если метод не используется — по умолчанию устанавливается автоматический выбор графы отбора.
Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ОбработкаДок)
|Период с '01.10.96' по '05.10.96';
|ОбрабатыватьДокументы Все;
|Тов = Документ.РасхНакл.Товар;
|Клиент = Документ.РасхНакл.Клиент;
|Группировка Клиент;
|Группировка Тов;
|Группировка Документ;
|"//)}ЗАПРОС
;
// ЕСЛИ ошибка в запросе, то выход из процедуры
Запрос.ИспользоватьГрафуОтбора("Клиент");
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Группировка

Получить следующее значение выборки запроса на заданном уровне группировки.
Синтаксис:
`Группировка(<Группировка>, <Направление>)`
Англоязычный синоним:
`Group`
Параметры:

`<Группировка>`	Выражение, содержащее порядковый номер или имя группировки.
`<Направление>`	Необязательный параметр. Числовое выражение: 1 — выборка значений группировки по возрастанию; -1 (минус единица) — выборка значений группировки по убыванию. Значение по умолчанию: 1.

Возвращаемое значение:
Число: 1 — если получено следующее значение выборки запроса, 0 — если нет.
Описание:
Метод `Группировка` служит для организации цикла получения данных из выборки, сформированной в результате работы метода `Выполнить`. Метод `Группировка` позиционирует в выборке очередную строку в порядке, определенном параметром `<Группировка>`. Нельзя использовать метод `Группировка`, задавая в качестве параметра младшие группировки, не использовав предварительно этот метод для позиционирования по старшим группировкам. Старшинство группировок определяется порядком их следования в тексте запроса.

Замечание: Обход группировок по многоуровневым справочникам в порядке убывания не выполняется.

Пример:

```
Процедура Сформировать()
   Перем Запрос, ТекстЗапроса;
   Если (Число(ДатаНач) = 0) ИЛИ (Число(ДатаКон) = 0) Тогда
      Предупреждение("Не задан период!");
      Возврат;
   КонецЕсли;
   //Создание объекта типа "Запрос" Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса =
   "//{{ЗАПРОС(Сформировать)
   |с ДатаНач по ДатаКон;
   |Рез = ЖурналРасчетов.Зарплата.Результат;
   |Сотр = ЖурналРасчетов.Зарплата.Объект;
```

```
    |Группировка Сотр без групп;
    |Группировка ПериодЖурнала;
    |Функция Сум = Сумма(Рез);
    |"//}}ЗАПРОС
    ;
    Если Сотрудник.Выбран() = 1 Тогда
       ТекстЗапроса = ТекстЗапроса + "Условие(Сотр = Сотрудник);";
    КонецЕсли;
    // Если ошибка в запросе, то выход из процедуры
    Если Запрос.Выполнить(ТекстЗапр) = 0 Тогда
       Возврат;
    КонецЕсли;
    // Подготовка к заполнению выходных форм данными запроса
    Таб = СоздатьОбъект("Таблица");
    // сначала используем группировку по сотрудникам
    // а потом группировку по периоду журнала!!!
    Пока Запрос.Группировка("Сотр") = 1 Цикл
       // Заполнение полей Сотр
       Таб.ВывестиСекцию("Сотр<");
       Пока Запрос.Группировка("ПериодЖурнала") = 1 Цикл
          // Заполнение полей ПериодЖурнала
          Таб.ВывестиСекцию("Мес");
       КонецЦикла;
       Таб.ВывестиСекцию("Сотр>");
    КонецЦикла;
    // Вывод заполненной формы
    // Заполнение полей "Итого"
    Таб.Опции(0, 0, 0, 0);
    Таб.ТолькоПросмотр(1);
    Таб.Показать("Результат", );
КонецПроцедуры
```

ЭтоГруппа

Возвращает флаг принадлежности группе справочника.

Синтаксис:

`ЭтоГруппа(<ИмяГруппировки>)`

Англоязычный синоним:

`IsItAGroup`

Параметры:

`<ИмяГруппировки>`	Выражение, содержащее порядковый номер или имя группировки (см. Главу «Язык запросов»).

Возвращаемое значение:

Число: 1 — если текущая строка выборки (временного набора данных, сформированного в результате выполнения запроса) является группой справочника, 0 — если это обычный элемент справочника.

Описание:

Метод `ЭтоГруппа` предназначен для выделения строк временного набора данных, которые являются группой справочника.

Пример:

```
Пока (Запр.Группировка("Товар") = 1) И (Запр.ЭтоГруппа("Товар") = 1) Цикл
   ...
КонецЦикла;
```

НачалоПериода

Возвращает дату начала периода формирования запроса.

Синтаксис:

`НачалоПериода()`

Англоязычный синоним:

`BeginOfPeriod`

Возвращаемое значение:

Значение типа «дата» — начало периода формирования запроса.

Описание:

Метод `НачалоПериода` возвращает дату начала периода запроса. Если в тексте запроса (см. «Язык запросов») указана одна из предопределенных группировок типа период (`"Год"`, `"Месяц"`, `"День"` и т. д.), то при обработке этой и вложенных в нее группировок метод `НачалоПериода` будет возвращать начало периодов текущих значений этих группировок.

Пример:

```
ДатаНачала = Запрос.НачалоПериода();
```

КонецПериода

Возвращает дату конца периода формирования запроса.

Синтаксис:

```
КонецПериода()
```

Англоязычный синоним:

```
EndOfPeriod
```

Возвращаемое значение:

Значение типа «дата» — конец периода формирования запроса.

Описание:

Метод КонецПериода возвращает дату конца периода запроса. Если в тексте запроса (см. «Язык запросов») указана одна из предопределенных группировок типа период ("Год" , "Месяц" , "День" и т. д.), то при обработке этой и вложенных в нее группировок метод КонецПериода будет возвращать конец периодов текущих значений этих группировок.

Пример:

```
ДатаКонца = Запрос.КонецПериода();
```

Получить

Прямое позиционирование на запись в выборке по конкретным значениям группировок.

Синтаксис:

```
Получить(<ЗначениеГруппировки_1>, ..., <ЗначениеГруппировки_n>)
```

Англоязычный синоним:

```
Get
```

Параметры:

<ЗначениеГруппировки_n>	Выражение, содержащее значение n-ой группировки запроса.

Возвращаемое значение:

Число: 1 — если запись найдена, 0 — если нет.

Описание:

Метод Получить осуществляет прямое позиционирование на запись в выборке по конкретным значениям группировок.

Количество параметров метода зависит от количества группировок в запросе. Можно пропускать значения одной или нескольких последних группировок, в этом случае метод позиционируется на запись, которая будет содержать итоговые значения для указанных группировок. Пропускаемые при вызове метода последние значения группировок должны заменяться запятыми. Если при вызове метода опущены все параметры, то выборка позиционируется на самое начало временного набора данных, на строку итогов.

После выполнения метода Получить может осуществляться дальнейший обход выборки вызовами метода Группировка.

Пример:

```
// Текст запроса
ТекстЗапроса = "
| ...
|Группировка Должность;
|Группировка Категория;
| ...
|";
// Текст процедуры обработки запроса
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
Если Запрос.Получить(ВыбДолжн, ВыбКатег) = 0 Тогда
   Если Запрос.Получить(ВыбДолжн, ) = 0 Тогда
      // ...
   КонецЕсли;
КонецЕсли;
```

ВНачалоВыборки

Осуществляет прямое позиционирование на начало выборки.

Синтаксис:

```
ВНачалоВыборки()
```

Англоязычный синоним:

```
ToSelectionBegin
```

Возвращаемое значение:

Число: 1 — если операция выполнена успешно, 0 — если нет.

Описание:

Метод ВНачалоВыборки осуществляет прямое позиционирование на начало выборки.

Если необходимо перейти на верхний уровень группировок, чтобы затем организовать проход группировки первого уровня (в любом направлении), следует использовать метод ВНачалоВыборки.

Пример:

```
// Текст запроса
ТекстЗапроса = "
| ...
|Группировка Должность;
|Группировка Категория;
| ...
|";
// Текст процедуры обработки запроса
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
Если Запрос.Получить(ВыбДолжн, ВыбКатег) = 0 Тогда
   Если Запрос.Получить(ВыбДолжн, ) = 0 Тогда
      // ...
   КонецЕсли;
КонецЕсли;
Запрос.ВНачалоВыборки();
Пока Запрос.Группировка(1, -1) цикл
   // ...
Конеццикла;
```

Выгрузить

Выгружает результаты запроса в таблицу значений.

Синтаксис:

```
Выгрузить(<ТаблЗнач>, <Флаг>, <Итоги>)
```

Англоязычный синоним:

```
Unload
```

Параметры:

`<ТаблЗнач>`	Таблица значений, куда выгружаются результаты запроса.
`<Флаг>`	Необязательный параметр. Число: 0 — значения групп и функций (по умолчанию); 1 — значения групп и функций, дополнительных переменных; 2 — значения упорядочиваний групп и функций; 3 — значения упорядочиваний групп и функций, дополнительных переменных; Строка — `"Товар(1), Товар(2), Товар, Склад, Приход, Расход"`, где Товар(1) — значение первого упорядочивания группировки `"Товар"`.
`<Итоги>`	Необязательный параметр. Число: 0 — итоги по группировкам не выводить; 1 — итоги по группировкам выводить сверху (по умолчанию); 2 — итоги по группировкам выводить снизу; 3 — итоги по группировкам выводить сверху и снизу.

Возвращаемое значение:

Число: 1 — если выгрузка произошла успешно, иначе — 0.

Описание:

Метод `Выгрузить` выгружает результаты запроса в таблицу значений.

Пример:

```
// Текст запроса
ТекстЗапроса = "
| ...
|Группировка Должность;
|Группировка Категория;
| ...
|";
// Текст процедуры обработки запроса
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
ТаблЗнач = Создать("ТаблицаЗначений");
Запрос.Выгрузить(ТаблЗнач, 0, 0);
```

ЗначениеУпорядочивания

Возвращает значение поля упорядочивания для заданной группировки.

Синтаксис:

```
ЗначениеУпорядочивания(<Группировка>, <Упорядочив>)
```

Англоязычный синоним:

```
OrderValue
```

Параметры:

<Группировка> Выражение, содержащее значение номера или имени группировки (номер работает быстрее).

<Упорядочив> Необязательный параметр. Порядковый номер параметра упорядочивания указанной группировки. Значение по умолчанию: 1.

Возвращаемое значение:

Значение поля упорядочивания.

Описание:

Метод ЗначениеУпорядочивания возвращает значение поля упорядочивания, выбирая его из временной выборки запроса, не обращаясь к базе данных.

Замечание: Если в тексте запроса для данной группировки не используется конструкция "Упорядочить по", то в этом случае действует упорядочивание по умолчанию:

- для справочников — зависит от основного представления при описании справочника в конфигураторе (код или наименование).
- для документов — дата и время создания документа.

Пример:

```
// Текст запроса
ТекстЗапроса = "
|ВидРаб = Документ.Наряд.ВидРаботы;
|Сотр = Документ.Наряд.Сотрудник;
|Группировка ВидРаб
|Упорядочить По ВидРаб.Код, ВидРаб.Стоимость;
|Группировка Сотр;
|";
//...
// Фрагмент заполнения таблицы
// Выбираем значение
ВидРаб.Стоимость
Запрос.ЗначениеУпорядочивания("ВидРаб", 2);
// Выбираем имя сотрудника
Запрос.ЗначениеУпорядочивания(2, 1);
```

ПолучитьАтрибут

Возвращает значение атрибута запроса по его имени.

Синтаксис:

ПолучитьАтрибут(<ИмяАтрибута>)

Англоязычный синоним;

SetAttrib

Параметры:

<ИмяАтрибута> Строковое выражение, содержащее значение любого атрибута запроса.

Возвращаемое значение:

Значение атрибута запроса.

Описание:

Метод ПолучитьАтрибут возвращает значение атрибута запроса по его имени.

Пример:

```
// Текст запроса
ТекстЗапроса = "
|ВидРаб = Документ.Наряд.ВидРаботы;
|Сотр = Документ.Наряд.Сотрудник;
|Группировка ВидРаб
|Упорядочить По ВидРаб.Код, ВидРаб.Стоимость;
|Группировка Сотр;
|";
// ...
// Фрагмент заполнения таблицы
// Выбираем значение ВидРаб
Запрос.ПолучитьАтрибут("ВидРаб");
// Выбираем сотрудника
Запрос.ПолучитьАтрибут("Сотр");
```

Глава 34
Язык Запросов

Язык запросов предназначен для описания запросов к документам, справочникам, регистрам, журналам расчетов, планам счетов, бухгалтерским операциям и проводкам. Текст описания запроса передается методу `Выполнить` (см. Главу «Работа с Запросами») в качестве параметра. В результате выполнения этого метода генерируется временный выходной набор данных, который в дальнейшем используется для заполнения формы отчета.

Формат текста описания запроса

Текст описания запроса на языке запросов состоит из последовательности операторов. Концом оператора является символ ";". Операторы могут записываться в любом порядке, однако, следует помнить, что интерпретатор языка запросов однопроходный, следовательно, сначала следует описать переменную, и только потом ее использовать в операторах `Группировка`, `Функция` или `Условие`.

Пример:

```
ТекстЗапроса =
"//({ЗАПРОС(РасчЛистки)
|  // Задаем интервал запроса
|Период с ДатаНач по ДатаКон;
|  // Определяем внутренние переменные
|Рез = ЖурналРасчетов.Зарплата.Результат;
|Расч = ЖурналРасчетов.Зарплата.ВидРасч;
|Дни = ЖурналРасчетов.Зарплата.Дни;
|Сотр = ЖурналРасчетов.Зарплата.Объект;
|  // Назначаем группировки
|Группировка Сотр без групп;
|Группировка Расч;
|  // Назначаем функции
|Функция Сум = Сумма(Рез);
|Функция Дней = Сумма(Дни);
|  // Назначаем условие
|Условие(Рез <> 0);
|"//}}ЗАПРОС
;
// ...
```

Соглашения и обозначения

В синтаксических диаграммах языка запросов используются следующие символы:

Символ	Значение
[ ]	В квадратных скобках заключаются необязательные синтаксические элементы.
[ ]$^{+}$	В квадратных скобках со знаком "+" заключаются обязательные синтаксические элементы, которые могут использоваться один или более раз.
[ ]*	В квадратных скобках со знаком "*" заключаются необязательные синтаксические элементы, которые могут использоваться один или более раз.
( )	Круглые скобки заключают в себе список параметров.
|	Вертикальной линией разделяются синтаксические элементы, среди которых нужно выбирать один и только один.

Зарезервированные слова языка запросов

Ключевые слова — это слова, которые используются языком запросов для обозначения встроенных операторов.

Приведенные далее ключевые слова являются зарезервированными и не могут использоваться в качестве имен внутренних переменных описания запросов. Каждое ключевое слово имеет как минимум 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 полный список зарезервированных слов языка запросов в обоих представлениях.

Русский яз.	Английский яз.
Без	Without
Год	Year
Групп	Groups
Группировка	Group
День	Day
Документ	Document
И	And
Или	Or

Квартал	Quarter
Когда	When
Месяц	Month
Неделя	Week
НомерСтроки	LineNum
Обрабатывать	Process
ОбрабатыватьДокументы	ProcessDocuments
ОбрабатыватьОперации	ProcessOperations
Период	Period
ПериодЖурнала	Journal Period
По	By
По	Till
С	From
СтрокаДокумента	Document Line
Упорядочить	Order
Условие	Condition
Функция	Function

Все ключевые слова могут быть набраны в любом регистре (верхний и нижний), например: `Функция`, `ФУНКЦИЯ`, `фУНКЦИЯ` .

Комментарий

Комментарий в тексте описания запроса начинается парой символов //и заканчивается концом строки.

Пример:

```
// Это — комментарий
Группировка Кат;  // Это тоже комментарий
```

Константы и переменные запросов

Константы

Язык содержит следующие типы констант:

Числовая константа описывается следующей синтаксической диаграммой:

$[+|-][0-9]^{+} \mid [0-9]^{*}.[0-9]^{+}$

Пример:

```
-17
43.712
.43842
```

Строковая константа — это строка, заключенная в двойные кавычки.

Пример:

```
"Это текстовая константа" // ОК
"Это ошибочная константа
и должна находиться на одной строке" // Ошибка
```

Константа типа "дата" задается в виде строки, заключенной в одинарные кавычки, в формате '`ДД.ММ.ГГ`' или '`ДД.ММ.ГГГГ`'.

Пример:

```
'21.05.96'
'25.09.1964'
```

Внутренние переменные

Внутренняя переменная — это переменная, объявленная в тексте описания запроса. Именем переменной может быть любая последовательность букв, цифр и знаков подчеркивания "_", начинающаяся с буквы или знака подчеркивания "_". Имена внутренних переменных не должны совпадать с зарезервированными словами языка запросов. Распознавание имен переменных, названий операторов, процедур и функций ведется без учета регистра букв.

Пример:

```
*
// имена внутренних переменных в описании запроса
_43842        // ОК
НачалоПериода // ОК
712piece      // Ошибка: начинается с цифры
Week          // Ошибка: ключевое слово зарезервировано
*
// применение имен внутренних переменных описания запроса
// в программном модуле после позиционирования строки
// выборки функцией Группировка
Сотр = НовЗапрос.Сотр;
```

```
Кат = НовЗапрос.Кат;
ИТОГ = НовЗапрос.Итого;
```

Конкретизация переменной

Конкретизация переменной — это уточнение описания внутренней переменной, если это возможно в текущем контексте. Конкретизации переменной могут использоваться в языке запросов в операторах `Группировка ... Упорядочить По` и в качестве аргумента оператора `Функция` (см. далее).

Синтаксис:

`<ВнутренняяПеременная>[.<Путь>]+;`

Параметры:

`<ВнутренняяПеременная>`	Идентификатор объявленной ранее внутренней переменной.
`<Путь>`	Доступный атрибут внутренней переменной или конкретизации переменной (см. «Атрибуты, доступные при описании внутренних переменных»).

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
" //{{ЗАПРОС(ОбработкаДок)
|Период с '01.10.96' по '05.10.96';
|ОбрабатыватьДокументы Все;
|Тов = Справочник.Товары.ТекущийЭлемент, Документ.РасхНакл.Товар;
|  // используем конкретизацию внутренней переменной Тов
|Группировка Тов Упорядочить По Тов.Наименование;
|Группировка Документ;
|Группировка СтрокаДокумента;
|"//}}ЗАПРОС
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Внимание! Запрос не манипулирует величинами типа «Строка неограниченной длины».

Внешние переменные

Внешние переменные — это переменные из области видимости процедуры или функции программного модуля, в теле которого используется данный запрос. Внешние переменные в тексте описания запроса могут использоваться в операторах `Период С` и `Условие`.

Пример:

```
Перем ДатаНач, ДатаКон;
Процедура Сформировать()
   Если (Число(ДатаНач) = 0) ИЛИ (Число(ДатаКон) = 0) Тогда
      Предупреждение("Не задан период!");
      Возврат;
   КонецЕсли;
   // Создание объекта типа "Запрос"
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса =
   "//{{ЗАПРОС(Сформировать)
   |  // используем внешние переменные ДатаНач и ДатаКон
   |с ДатаНач по ДатаКон;
   |Рез = ЖурналРасчетов.Зарплата.Результат;
   |Сотр = ЖурналРасчетов.Зарплата.Объект;
   |Группировка Сотр без групп;
   |Группировка ПериодЖурнала;
   |Функция Сум = Сумма(Рез);
   |"//}}ЗАПРОС
   ;
   // используем внешнюю переменную ВыбСотр
   Если ВыбСотр.Выбран() = 1 Тогда
      // используем в запросе внешнюю переменную ВыбСотр
      ТекстЗапроса = ТекстЗапроса + "Условие(Сотр = ВыбСотр);";
   КонецЕсли;
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапр) = 0 Тогда
      Возврат;
   КонецЕсли;
   ...
КонецПроцедуры // ВыбСотр — это реквизит диалога
```

Атрибуты, доступные при описании внутренних переменных

В языке запросов можно обращаться к атрибутам следующих видов данных:

Русское назв.	**Англ. Назв.**	**Описание**
Документ	Document	Данные документов.
Справочник	Reference	Данные справочников.
Регистр	Register	Данные регистров.
ЖурналРасчетов	CalcJournal	Данные журналов расчета.
Счет	Account	Данные счетов.
Операция	Operation	Данные бухгалтерских операций и проводок.

Эти названия являются первыми в пути описания переменных. В качестве атрибутов допускается использовать любые реквизиты, которые заданы для них в конфигураторе в дереве метаданных (реквизиты для справочников, документов и журналов расчетов; измерения и ресурсы для регистров). Кроме этих атрибутов, разрешен так же доступ к следующим предопределенным атрибутам:

Доступные атрибуты объектов типа «Документ»:

Русское назв.	**Англ. Назв.**	**Описание**
ВремяДок	DocTime	Время документа.
ДатаДок	DocDate	Дата документа.
НомерДок	DocNum	Номер документа.
НомерСтроки	LineNum	Номер строки документа.
ТекущийДокумент	CurDocument	Значение текущего документа.

Доступные атрибуты объектов типа «Справочник»:

Русское назв.	**Англ.назв.**	**Описание**
родитель	Parent	Родитель элемента многоуровневого справочника.
Владелец	Owner	Владелец подчиненного справочника.
Код	Code	Код элемента справочника.
Наименование	Description	Наименование элемента справочника.
ТекущийЭлемент	CurItem	Значение текущего элемента справочника.

Доступные атрибуты объектов типа «Регистр»:

Русское назв.	**Англ.назв.**	**Описание**
НомерСтроки	LineNum	Номер строки документа, выполнившего движение регистра (в случае, когда в Модулях документов в конфигурации перед движением регистра использовали метод ПривязыватьСтроку).
ТекущийДокумент	CurDocument	Значение документа, выполнившего движение регистра.

Доступные атрибуты объектов типа «ПланСчетов»:

Русское назв.	**Англ. назв.**	**Описание**
БезКорреспонденций	IsSingle	Флаг того, что элемент плана счетов — забалансовый.
Валютный	IsCurrency	Флаг валютного учета элемента плана счетов.
КОД	Code	Код элемента плана счетов.
Количественный	Is Amount	Флаг количественного учета элемента плана счетов.
Наименование	Description	Наименование элемента плана счетов.
ТекущийСчет	CurAccount	Значение текущего счета плана счетов.

Доступные атрибуты объектов типа «Операция»:

Русское назв.	**Англ. Назв.**	**Описание**
ВремяОперации	OperTime	Время операции.
ДатаОперации	OperDate	Дата операции.
Содержание	Description	Содержание операции.
СуммаОперации	OperSum	Сумма операции.
ТекущийДокумент	CurDocument	Значение документа создавшего операцию.

Доступные атрибуты объектов типа «Проводка»:

Русское назв.	**Англ. назв.**	**Описание**
ВалСумма	CurSum	Валютная сумма проводки (для счетов с валютным учетом).
Валюта	Currency	Валюта проводки.
Количество	Amount	Количество проводки (для счетов с количественным учетом).
КорСчет	CorAccount	Корреспондирующий счет. Корреспондирующим счетом, для которого является Счет.
Сумма	Sum	Сумма проводки.
Счет	Account	Счет, для обработки корреспонденции. Корреспондирующим счетом, для которого является КорСчет.

Пример:

```
*
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
|Период с НачДата по КонДата;
|ДОКУМ = Документ.РасходнаяБН.ТекущийДокумент,
|   Документ.РасходнаяКредит.ТекущийДокумент,
|   Документ.РасходнаяНал.ТекущийДокумент,
|   Документ.РасходнаяРеализ.ТекущийДокумент,
|   Документ.Счет.ТекущийДокумент;
|Группировка ДОКУМ; //по документам
|"//}}ЗАПРОС
;
*
//Создание объекта типа Запрос
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Печать)
|Товар = Справочник.Товары.ТекущийЭлемент;
|Группировка Товар Упорядочить по Товар.МинЗапас;
|"//}}ЗАПРОС
*
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|Период с НачДата по КонДата;
|ВидТоплива = Регистр.ПокупателиКолво.ВидыТоплива;
|Вес = Регистр.ПокупателиКолво.Кг;
|Покуп = Регистр.ПокупателиКолво.Покупатели;
|Док = Регистр.ПокупателиКолво.ТекущийДокумент;
|Ном = Регистр.ПокупателиКолво.НомерСтроки;
|Группировка ВидТоплива; //по измерению Регистра
|Группировка Док; // по документам, двигавшим Регистр
|Группировка Ном; // по номерам строк документов
|Функция ВсегоКолво = КонОст(Вес);
|Функция ПриходКолво = Приход(Вес);
|Условие(Покуп = ВыбПокупатель);
|"//}}ЗАПРОС
;
```

Операторы языка запросов

Объявление внутренней переменной

Синтаксис:

<ИмяПеременной> = <ОписаниеПеременной> [, <ОписаниеПеременной>]+;

Параметры:

<ИмяПеременной>	Имя объявляемой внутренней переменной описания запроса.
<ОписаниеПеременной>	Указывает на доступный в языке запросов атрибут документа, справочника, регистра или журнала расчетов (см. «Атрибуты, доступные при описании внутренних переменных»).

Описание:

Внутренние переменные используются в тексте запроса для образования ссылок на объекты конфигурации, чтобы использовать их при построении таких операторов запроса, как Группировка, Функция, Условие.

В объявлении внутренней переменной можно указывать несколько вариантов <ОписанияПеременной>. Все описания должны указывать на один и тот же тип данных (число, строку, справочник или документ). Переменной, указывающей на разные справочники или документы присваивается тип данных «Справочник неопределенного вида» или «Документ неопределенного вида» соответственно.

* Например, можно определить внутреннюю переменную:

```
Товар = Документ.Перемещение.Товар, Документ.Расходная.Товар;
```

и использовать ее для создания Группировки. Такой состав внутренней переменной как бы дает Группировке задание — просмотреть все документы видов "Перемещение" и "Расходная" и выбрать все товары, встречающиеся либо в документах вида "Перемещение", либо в документах "Расходная".

* Еще пример. Допустим, у нас есть регистр "Взаиморасчеты" и регистр "Кредиты", и в том и в другом есть измерение "Клиент". Определяем внутреннюю переменную:

```
Должник = Регистр.Взаиморасчеты.Клиент, Регистр.Кредиты.Клиент;
```

Если использовать такую внутреннюю переменную для образования Группировки, то она будет означать следующее: пройти по регистру "Взаиморасчеты" и по регистру "Кредиты", вычислить заданные в запросе Функции и выбрать значения "Клиентов", для которых значения хотя бы одной функции будет ненулевой. Здесь мы видим,

что формирование запроса по регистрам имеет некоторые особенности (обязательно наличие Функций, причем их вычисленные значения должны быть отличны от нуля, только в этом случае найденные объекты включаются во временный набор данных формируемый запросом).

Если описания внутренней переменной указывают на разные типы данных, например, на «Справочник» и на «Документ», то это просто вызовет сообщение об ошибке. Например, следующее определение внутренней переменной будет ошибочным:

```
ААА = Справочник.Товары.ТекущийЭлемент, Документ.Счет.ТекущийДокумент;
// ЭТО ОШИБКА !!!
```

Однако, допускается в описании внутренней переменной указание на разные справочники либо на разные документы. Переменной, указывающей на разные справочники или документы присваивается тип данных «Справочник неопределенного вида» или «Документ неопределенного вида» соответственно.

* Например, правомерно задать такую внутреннюю переменную:

```
Парам = Документ.Счет.Клиент, Документ.Счет.Фирма;
```

(Здесь предполагается, что в контексте конфигурации, реквизиты "Клиент" и "Фирма" — это элементы справочников). Если использовать такую внутреннюю переменную для построения Группировки, то она будет означать следующее: пройти по документам "Счет" и выбрать значения клиентов и фирм, встречающихся в них (и, например, упорядочить эти элементы по наименованию). Как интерпретировать результаты полученной выборки при таком запросе — будет зависеть от контекста поставленной задачи.

В языке запросов описание внутренних переменных для агрегатных типов данных типа «Операция» и «Проводка» начинается со слова «Операция» вне зависимости от того, будет ли обращение к бухгалтерским операциям или к проводкам. <ОписаниеПеременной> определяется следующим образом:

```
Операция.(
   <Реквизиты0перации>
   |  <ПредРеквОпераций>
   |  <РеквизитыПроводок>
   |  <ПредРеквПроводок>
   |  Дебет.(
         Счет
         | <Субконто>
      )
   |  Кредит.(
         Счет
         | <Субконто>
      )
   |  Субконто.
      <ВидСубконто>
   |  КорСубконто.
      <ВидСубконто>
)
```

<РеквизитыОперации>	Реквизиты операции объявленные в метаданных.
<ПредРеквОпераций>	Предопределенные реквизиты Операций.
<РеквизитыПроводок>	Реквизиты проводок, объявленные в метаданных.
<ПредРеквПроводок>	Предопределенные реквизиты Проводок.
<ВидСубконто>	Идентификатор вида субконто.

Применение слов "Дебет", "Кредит" в описании переменных позволяют оперировать дебетовой и кредитовой частями проводки, такими как счет и вид субконто.

Применение слов "Субконто" и "КорСубконто" в описании переменных позволяют оперировать как субконто, так и корреспондирующим субконто.

*

```
Сум = Операция.Сумма;
Сч = Операция.Дебет.Счет, Операция.Кредит.Счет;
```

Кратко суть этого раздела можно выразить так: при определении внутренних переменных запроса разработчик конфигурации должен ясно себе представлять суть и смысл взаимосвязи объектов конфигурации, которые объединяются в описании единой внутренней переменной, поскольку интерпретация полученных результатов выполнения запроса полностью зависит от контекста решаемой задачи.

Несколько слов о дополнительных именах доступа к стандартной информации, которые определены в языке запросов. Речь идет о конструкции ТекущийЭлемент для справочников и ТекущийДокумент для документов и регистров (не надо путать эти конструкции языка запросов с одноименными методами встроенного языка, хотя их смысл во многом совпадает).

Для справочников использование конструкции ТекущийЭлемент в описании внутренней переменной означает выборку элемента справочника как такового (целиком всей записи справочника).

Аналогично при работе с документами, использование конструкции ТекущийДокумент в описании внутренней переменной, означает выборку документа как такового (целиком всего документа как объекта конфигурации). Сравните два примера:

Пример 1.

* Здесь нас интересуют товары, встречающиеся в документах.

```
ТОВАР = Документ.РасходнаяБН.Товар,
```

```
   Документ.РасходнаяКредит.Товар,
   Документ.РасходнаяНал.Товар,
   Документ.РасходнаяРеализ.Товар,
   Документ.Счет.Товар;
```

Пример 2.

* Здесь нас интересуют сами документы указанных видов как таковые.

```
ДОКУМ = Документ.РасходнаяБН.ТекущийДокумент,
   Документ.РасходнаяКредит.ТекущийДокумент,
   Документ.РасходнаяНал.ТекущийДокумент,
   Документ.РасходнаяРеализ.ТекущийДокумент,
   Документ.Счет.ТекущийДокумект;
```

Использование для документов конструкции `НомерСтроки` в описании внутренней переменной, означает выборку номеров строк документов (не самих строк, а чисел, обозначающих номера строк в документе).

```
СТРОКА = Документ.РасходнаяБН.НомерСтроки;
```

Для регистров использование конструкции `ТекущийДокумент` в описании внутренней переменной означает выборку документов (целиком всего документа как объекта конфигурации), которые произвели движение по данному регистру.

Использование для регистра конструкции `НомерСтроки` в описании внутренней переменной, означает выборку связанных номеров строк тех документов, которые произвели движение по регистру (в случае, когда в Модулях документов в конфигурации перед движением регистра использовали метод `ПривязыватьСтроку`).

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|Период с НачДата по КонДата;
|ВидТоплива = Регистр.ПокупателиКолво.ВидыТоплива;
|Покуп = Регистр.ПокупателиКопво.Покупатели;
|Док = Регистр.ПокупателиКолво.ТекущийДокумент;
|Ном = Регистр.ПокупателиКолво.НомерСтроки;
|Вес = Регистр.ПокупателиКолво.Кг;
|Группировка ВидТоплива; // по измерению Регистра
|Группировка Док; //по документам, двигавшим Регистр
|Группировка Ном; // по номерам строк документов
|Функция ВсегоКолво = КонОст(Вес);
|Функция ПриходКолво = Приход(Вес);
|Условие(Покуп = Покупат);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

В программном модуле конфигурации (т. е. вне текста запроса) существует доступ (там, где это имеет смысл) к значениям всех объявленных в тексте запроса внутренних переменных, даже тех, которые не участвуют ни в группировках, ни в функциях. Другими словами, если объявленная в тексте запроса внутренняя переменная не задействована в запросе ни в группировке, ни в функции ни в условии, данная внутренняя переменная все же присутствует в запросе.

Период С

Установить временной интервал дат формирования запроса.

Синтаксис:

```
[[Период] С <Дата>|<ВнешняяПерем> [ПО <Дата>|<ВнешПеременная>];]
```

Англоязычный синоним:

```
[[Period] From <Дата>|<ВнешПеременная> [Till <Дата> | <ВнешПеременная>];]
```

Параметры:

`<Дата>`	Константа типа «Дата», «Документ» или . позиция документа.
`<ВнешПеременная>`	Внешняя переменная типа «Дата», «Документ» или позиция документа. Если указан документ, то за момент времени принимается дата и время документа.
`По`	Добавочное ключевое слово для связки первой части команды со второй, необязательной частью.

Описание:

Оператор `Период С` устанавливает интервал дат формирования запроса. Если в описании запроса оператор `Период С` опущен, то интервал дат формирования запроса устанавливается в *точку актуальности* итогов (ТА) (или на `РабочуюДату`, если не установлена компонента «Оперативный учет»), запрос формируется только на этот момент времени.

Использование данного оператора актуально только в том случае, если запрос строится для выборки данных по регистрам, журналам расчетов и/или документам. Здесь указывается, в каком интервале выбирать движения регистров и/или подборку документов. При выполнении запроса только по справочникам эта секция не играет никакой роли.

Параметрами оператора Период С являются значения момента времени («Дата», «Документ» или позиция документа) начала и конца временного интервала. Следует особо обратить внимание, что если интервал задается с точностью до даты, то интервал времени считается от начала даты нижней границы интервала до конца даты верхней границы интервала. Если вторая часть оператора после ключевого слова По пропущена или значение второго параметра команды равно нулю, то интервал времени применяется от начального момента времени до ТА (или по РабочуюДату, если не установлена компонента «Оперативный учет»). Это особенно важно при формировании запросов по регистрам, т. к. запрос по регистрам может строиться от любой даты в прошлом до ТА. Поэтому, если вы укажете в запросе верхнюю границу интервала большей или равной дате ТА, то программа скорее всего сообщит «Не могу рассчитать регистры за ТА» (поскольку время ТА лежит где то в пределах даты, а запрос пытается учесть всю дату в целом). Поэтому при формировании текста запроса следует вставлять дополнительную проверку типа той, что приведена в следу ющем примере:

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"Счет = Регистр.Cash.Счет;
|Статья = Регистр.Cash.Статья;
|СуммаЭквивалента2 = Регистр.Cash.СуммаЭквивалента2;
|СуммаЭквивалента1 = Регистр.Cash.СуммаЭквивалента1;
......
Если ДатаКонца >= ПолучитьДатуТА() Тогда
   ТекстЗапроса = ТекстЗапроса + "Период с ДатаНачала;";
Иначе
   ТекстЗапроса = ТекстЗапроса + "Период с ДатаНачала по ДатаКонца;";
КонецЕсли;
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Работа команды при формировании запросов к журналам расчетов зависит от того, используется ключевое слово "Период" или нет. Применение конструкции "Период С ... По..." приводит к тому, что выборка записей из журнала расчетов осуществляется в разрезе расчетных периодов конкретного журнала, фактически, по времени ввода строк журнала расчетов в систему.

Применение конструкции "С ... По ..." приводит к тому, что выборка записей из журнала расчетов проводится по времени действия записей журнала расчетов, определяемых реквизитами журнала расчетов "ДатаНачала" и "ДатаОкончания", а не временем их ввода в журнал. Более подробно о двойственности временного представления журналов расчетов см. выше, в главе «Работа с Журналами расчетов».

Замечание. Работа запроса с журналами расчетов, с использованием конструкции "Период С ... По ...", аналогична выборке записей в журнале расчетов, организуемой при помощи метода журнала расчетов ВыбратьПериод. С другой стороны, запрос, использующий конструкцию "С ... По ...", аналогичен выборке, организуемой при помощи метода журнала расчетов ВыбратьЗаписи.

Пример:

```
*
Процедура РасчЛистки()
   Перем Запрос, Флаг;
   Перем ДатаНач, ДатаКон;
   ЖР = СоздатьОбъект{"ЖурналРасчетов.Зарплата");
   Нач = ЖР.НачалоТекущегоПериода();
   Кон = ЖР.КонецТекущегоПериода();
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   Флаг = Запрос.Выполнить(
   "//{{ЗАПРОС(РасчЛистки)
   |Период С ДатаНач По ДатаКон; // выборка по расчетным периодам!
   |Рез = ЖурналРасчетов.Зарплата.Результат;
   |Расч = ЖурналРасчетов.Зарплата.ВидРасч;
   |Дни = ЖурналРасчетов.Зарплата.Дни;
   |Сотр = ЖурналРасчетов.Зарплата.Объект;
   |Группировка Сотр без групп;
   |Группировка Расч;
   |функция Сум = Сумма(Рез);
   |функция Дней = Сумма(Дни);
   |Условие(Рез о 0);
   |"//}}ЗАПРОС
   );
   // ...
КонецПроцедуры;
```

```
*
//Создание объекта типа Запрос
Запрос = СоздатьОбъект("Запрос");
// ниже формируется текст запроса с выборкой по времени
// действия записей журнала расчетов, а не по расчетным периодам!
ТекстЗапроса =
"//{{ЗАПРОС(Сформировать)
|С ДатаНач По ДатаКон;
|Рез = ЖурналРасчетов.Зарплата.Результат;
|Сотр = ЖурналРасчетов.Зарплата.Объект;
|ПЖ = ЖурналРасчетов.Зарплата.ПериодРегистрации;
|Группировка Сотр без групп;
|Группировка ПЖ;
|Функция Сум = Сумма(Рез);
"//}}ЗАПРОС
;
Флаг = Запрос.Выполнить(ТекстЗапроса);
// ...
```

ОбрабатыватьДокументы

Назначить условие обработки документов в запросе.

Синтаксис:

```
ОбрабатыватьДокументы[Непроведенные|Проведенные|Все];
```

Англоязычный синоним:

```
ProcessDocuments[NonTransacted|Transacted|All];
```

Описание:

Оператор `ОбрабатыватьДокументы` назначает режим обработки документов в запросе. В операторе указывается, какими документами должен оперировать запрос: проведенными, непроведенными или теми и другими. По умолчанию в запросе обрабатываются только проведенные документы.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "//{{ЗАПРОС(ОбработкаДок)
|Период с '01.10.96' по '05.10.96';
|ОбрабатыватьДокументы Все;
|Тов = Справочник.Товары.ТекущийЭлемент, Документ.РасхНакл.Товар;
|Группировка Тов упорядочить по Тов.Наименование;
|Группировка Документ;
|Группировка СтрокаДокумента;
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

ОбрабатыватьОперации

Назначить условие обработки бухгалтерских операций в запросе.

Синтаксис:

```
ОбрабатыватьОперации[Включенные|Выключенные|Все];
```

Англоязычный синоним:

```
ProcessOperations[On|Off|All];
```

Описание:

Оператор `ОбрабатыватьОперации` назначает режим обработки бухгалтерских операций в запросе. В операторе указывается, какими операциями должен оперировать запрос: с включенными проводками, с выключенными проводками или теми и другими. По умолчанию в запросе обрабатываются только операции с включенными проводками.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(ОбработкаОпер)
|Период с '01.10.96' по '05.10.96';
|ОбрабатыватьОперации Все;
|Опер = Операция.ТекущийДокумент;
|Группировка Опер;
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
```

```
КонецЕсли;
```

Обрабатывать

Назначить условие обработки помеченных на удаление объектов в запросе.

Синтаксис:

```
Обрабатывать[ПомеченныеНаУдаление|НеПомеченныеНаУдаление|Все];
```

Англоязычный синоним:

```
Process[MarkedOnRemoving|NonMarkedOnRemoving|All];
```

Описание:

Оператор `Обрабатывать` назначает режим обработки помеченных на удаление объектов в запросе. В операторе указывается, какими объектами должен оперировать запрос: не помеченными на удаление, помеченными на удаление или теми и другими. По умолчанию в запросе обрабатываются все объекты.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Обработка)
|Период с '01.10.96' по '05.10.96';
|Обрабатывать НеПомеченныеНаУлаление;
|Товар = Справочник.Товар.ТекущийЭлемент;
|Группировка Товар упорядочить по Товар.Наименование;
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Функция

Вычисляет функцию и включает ее результат в запрос.

Синтаксис:

```
Функция <ИмяФункции> = <ТипФункции> (<Параметр>|<УточненныйПараметр>)
                  [Когда (<Условие>)];
```

Англоязычный синоним:

```
Function <ИмяФункции>= <ТипФункции> (<Параметр>|<УточненныйПараметр>)
                  [When (<Условие>)];
```

Параметры:

`<ИмяФункции>`	Имя, которое присваивается функции. По этому имени можно в дальнейшем обращаться к значению вычисленной функции из вызывающего программного модуля как к атрибуту запроса. Данную внутреннюю переменную заранее объявлять не нужно. Она фактически неявно объявляется в операторе `Функция`.
`<ТипФункции>`	Ключевое слово одной из встроенных функций языка. Может быть одним из приведенных ниже.
`<Параметр>`	Имя объявленной ранее внутренней переменной, значение которой используется как параметр встроенной функции `<ТипФункции>`. В функциях: `Сумма`, `Среднее`, `Максимум`, `Минимум` в качестве данного параметра возможно использование арифметического выражения в терминах встроенного языка 1С:Предприятие.
`<УточненныйПараметр>`	Конкретизация объявленной ранее внутренней переменной, значение которой используется как аргумент встроенной функции < `ТипФункции`>.
`Когда`	Необязательное добавочное ключевое слово, использование которого в команде означает, что вычисление функции следует производить только при условии, когда заданное логическое выражение `<Условие>` истинно.
`<Условие>`	Логическое выражение встроенного языка 1С:Предприятие. В логическом выражении могут участвовать как внутренние, так и внешние переменные запроса (см. описание метода `Условие`). Используется только после ключевого слова `Когда`.

Описание:

Оператор `Функция` вычисляет функцию `<ТипФункции>` и присваивает ее значение внутренней переменной `<ИмяФункции>`.

Типы применяемых функций предопределены и могут быть следующими:

Тип Функции	Англояз. Синоним	Выполняемое действие
`Сумма`	`Sum`	Вычисляет сумму выбранных по запросу значений параметра.[1]
`Среднее`	`Avg`	Вычисляет среднее из выбранных по запросу значений параметра.
`Минимум`	`Min`	Вычисляет минимум из выбранных по запросу значений параметра.
`Максимум`	`Max`	Вычисляет максимум из выбранных по запросу значений параметра.
`Счётчик`	`Count`	Подсчитывает количество записей, вошедших в выборку.
`НачОст`	`BegRest`	Вычисляет начальный остаток для выбранных по запросу значений параметра.[2]

КонОст	EndRest	Вычисляет конечный остаток для выбранных по запросу значений параметра.[2]
Приход	Debit	Вычисляет приход для выбранных но запросу значений параметра.[2]
Расход	Credit	Вычисляет расход для выбранных по запросу значений параметра.[2]
СНД	IDB	Вычисляет сальдо начальное дебетовое для выбранных по запросу значений параметра.[3]
СКД	FDB	Вычисляет сальдо конечное дебетовое для выбранных по запросу значений параметра.[3]
СНК	ICD	Вычисляет сальдо начальное кредитовое для выбранных по запросу значений параметра.[3]
СКК	FCD	Вычисляет сальдо конечное кредитовое для выбранных по запросу значений параметра.[3]
ДО	TD	Вычисляет дебетовые обороты для выбранных по запросу значений параметра.
КО	TC	Вычисляет кредитовые обороты для выбранных по запросу значений параметра.[3]
КорДО	CorTD	Вычисляет дебетовые обороты между корреспондирующим счетам или субконто для выбранных по запросу значений параметра.[3, 4]
КорКО	CorTC	Вычисляет кредитовые обороты между корреспондирующим счетам или субконто для выбранных по запросу значений параметра.[3, 4]

[1]**Замечание**: Для ресурсов оборотных регистров допускается вызывать только тип функции "Сумма".

[2]**Замечание**: Типы функций НачОст, КонОст, Приход, Расход можно использовать только с параметрами, указывающими на ресурсы регистров остатков. Для ресурсов регистров остатков другие типы функций вызывать нельзя.

[3]**Замечание**: Функции СНД, СНК, СКД, СКК, ДО, КО, КорДО, КорКО можно использовать только с реквизитами проводки: Сумма, Количество или ВалСумма.

[4]**Замечание**: Функции КорДО и КорКО накапливают значения только тогда когда в запросе есть обращение к реквизитам проводок Счет, КорСчет Субконто или КорСубконто.

Замечание: В программном модуле, где используется запрос, <ИмяФункции> будет являться атрибутом запроса. При помощи данного атрибута можно обращаться к значению вычисленной в запросе функции.

Замечание: В функциях: Сумма, Среднее, Максимум, Минимум в качестве аргумента возможно использование арифметического выражения в терминах встроенного языка.

Пример:

```
...
|КолВо = Документ.ВидДокумента.Количество;
|Цена = Документ.ВидДокумента.Цена;
|Функция Сум = Сумма(КолВо * Цена);
|Функция Макс = Максимум(Окр(КолВо) * Окр(Цена));
|Функция Средн = Среднее(ФункцияОпределеннаяВМодуле(КолВо, Цена));
...
```

Пример:

```
// Создание объекта типа Запрос
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Функции)
|Период с ДатаНач по ДатаКон;
|Оклад = Справочник.Сотрудники.Оклад;
|Подр = Справочник.Сотрудники.Подразделение;
|Ктг = Справочник.Сотрудники.Категория;
|Группировка Подр без групп;
|Группировка Ктг без групп;
|Функция Всего = Сумма(Оклад);
|Условие((Ктг.Выбран() = 1) И (Подр.Выбран() = 1));
|"//}}ЗАПРОС
;
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
// ...
Итог = Запрос.Всего;
// ...
```

Оператор Функция предназначен для вычисления и накопления некоторых численных значений в процессе формирования выборки по запросу.

Кроме того, при формировании временного набора данных запрос специально формирует итоговые записи, в которые заносит накопленные значения вычисленных функций для каждой вложенной Группировки (подробнее об этом см. главу данной книги «Структура запросов и методика их использования»).

Операторы функций в запросе можно опускать, тогда запрос просто не будет ничего вычислять, а просто во временный набор данных заносятся записи со значениями Группировок. Однако это справедливо только для работы запросов по справочникам и по документам. При работе запроса по регистрам следует помнить, что в этом случае обязательно наличие Функций, причем только в том случае, когда их вычисленные значения отличны от нуля (хотя бы одной из заданных Функций), найденные объекты включаются во временный набор данных, формируемый запросом. Если при работе запроса по регистрам в тексте запроса операторы функций опущены, то программа не выдаст никакого сообщения об ошибке, добросовестно пройдет по всем Группировкам, не вычислит никаких значений Функций и, значит, ничего не запишет во временном файле.

При создании в тексте запроса внутренней переменной, которую вы предполагаете использовать в качестве параметра оператора Функция, надо учитывать, что тип этой внутренней переменной должен быть «число», т. к. функции в языке запросов обрабатывают только численные значения.

В тексте запроса, при описании оператора `Функция` можно использовать необязательное ключевое слово `Когда`, использование которого в операторе означает, что вычисление функции следует производить только при условии, что значение логического выражения, заданного в параметре ключевого слова является ИСТИНА. Синтаксис применяемого логического выражения полностью аналогичен синтаксису разрешенному к применению в операторах `Условие`.

Следует понимать, что не все функции внутри конкретной группировки запроса могут иметь четко интерпретируемый смысл. Например, для группировки по документу движения регистра следующие функции

```
|Функция ПрихКол = Приход(Количество);
|Функция РасхКол = Расход(Количество);
```

имеют четкий смысл — приращения, сделанные документом при движении регистра. С другой стороны, в той же группировке следующие функции:

```
|Функция НачКол = НачОст(Количество);
|Функция КонКол = КонОст(Количество);
```

явно не имеют смысла (в запросах по регистрам, обычно задают период запроса при помощи оператора `Период С`. Функция `НачКол` в данном примере должна по смыслу показывать остаток ресурса "`Количество`" на начальную дату запроса. Внутри группировки по документу вопрос: «Какой начальный остаток ресурса на дату 10.01.97?» по документу, проведенному, например, 13.01.97, не имеет смысла). Поэтому в таких ситуациях функция будет иметь нулевое значение.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса=
'//{{ЗАПРОС(Функции)
|Период с ДатаОтчета;
|Товар = Регистр.КвотыТоваров.Товар;
|Отдел = Регистр.КвотыТоваров.Отдел;
|Сотрудник = Регистр.КвотыТоваров.Сотрудник;
|Партнер = Регистр.КвотыТоваров.Партнер;
|Док = Регистр.КвотыТоваров.ТекущийДокумент;
|Количество = Регистр.КвотыТоваров.КвотаТовара;
|Группировка Товар;
|Группировка Отдел;
|Группировка Сотрудник;
|Группировка Партнер;
|Группировка Док;
|Функция НачКол = НачОст(Количество);
|Функция ПрихКол = Приход(Количество);
|Функция РасхКол = Расход(Количество);
|Функция КонКол = КонОст(Количество);
|// Следующие Функции вычисляем только при определенных
|// условиях, чтобы получить отфильтрованные итоги
|Функция ПрихКолТов = Приход(Количество) Когда (Док.ФлагТовара = 1);
|Функция РасхКолТов = Расход(Количество) Когда (Док.ФлагТовара = 1);
|Функция ПрихКолОтд = Приход(Количество) Когда (Док.ФлагОтдела = 1);
|Функция РасхКолОтд = Расход(Количество) Когда (Док.ФлагОтдела = 1);
|Функция ПрихКолСотр = Приход(Количество) Когда (Док.ФлагСотрудника = 1);
|Функция РасхКолСотр = Расход(Количество) Когда (Док.ФлагСотрудника = 1);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
   Возврат;
КонецЕсли;
```

Группировка

Устанавливает порядок выборки информации.

Синтаксис:

```
Группировка <ИмяГруппировки>|<ПредопредГруппировка>
            [Упорядочить по <Порядок>[, <Порядок>]*]
            [Без Упорядочивания][Без Групп][Все [ВошедшиеВЗапрос]];
```

Англоязычный синоним:

```
Group <ИмяГруппировки>|<ПредопредГруппировка>
            [Order By <Порядок>[, <Порядок>]*]
            [Without Order][Without Groups][All [IncludedInQuery]];
```

Параметры:

<ИмяГруппировки>	Имя объявленной ранее внутренней переменной, по значению которой устанавливается порядок выборки. По этому имени можно в дальнейшем обращаться к значению группировки из вызывающего программного модуля как к атрибуту запроса.
<ПредопредГруппировка>	Ключевое слово одной из встроенных предопределенных группировок языка запросов. По этому имени можно будет обращаться к значению группировки из вызывающего программного модуля. Возможные значения приведены ниже.
упорядочить по	Необязательное добавочное ключевое слово. Параметры, следующие за данным ключевым словом, определяют упорядочивание строк в группировке. По умолчанию документы упорядочиваются по дате и времени документов, справочники — по коду или наименованию, в зависимости от основного представления, заданного при описании справочника в конфигураторе.
<Порядок>	Используется только после ключевого слова Упорядочить по. Конкретизация внутренней переменной <ИмяГруппировки>, значение которой является параметром упорядочивания строк в группировке. Кроме того, в данном параметре можно использовать имя функции, объявленной в этом же запросе в операторе Функция.
Без Упорядочивания	Необязательное добавочное ключевое слово, которое преследует цель уменьшения времени формирования запроса, при условии, что ни упорядочивание, ни значения упорядочивания при использовании данного запроса не нужны.
Без Групп	Необязательное добавочное ключевое слово, использование которого назначает вывод в запрос только простых элементов справочников (исключая группы). Используется только для группировок, построенных на основе внутренней переменной типа «справочник».
Все	Необязательное добавочное ключевое слово, действие которого зависит от типа внутренней переменной, на основе которой построена группировка. Используется только для группировок, построенных на основе внутренней переменной типа «справочник» или для предопределенных группировок, задающих временной интервал (Год, Месяц, Квартал, Неделя, День). Для предопределенных временных группировок подразумевается, что в запрос будут включены любые значения данных (в том числе нулевые) в каждый заданный момент времени с даты начала запроса по дату конца запроса (интервал задается оператором Период С...) Для группировок по справочникам подразумевается, что в запрос будут включены любые значения данных (в том числе нулевые) для каждого допустимого элемента справочника.
ВошедшиеВЗапрос	Необязательное добавочное ключевое слово действие которого уточняет предыдущее ключевое слово «Все». Использование данного слова подразумевает, что в каждую строку запроса будут включены значения данных (в том числе нулевые), для которых есть ненулевое значение хотя бы в одной строке запроса.

Описание:

Оператор Группировка задает способ группировки информации и порядок ее упорядочивания в запросе.

Предопределенные группировки могут быть следующими:

Предопределенн. Группировка	Англояз. Синоним	Выполняемое действие
Документ	Document	Позволяет детализацию до каждого документа.
СтрокаДокумента	DocumentLine	Позволяет детализацию до каждой строки табличной части каждого документа.
День	Day	Группировки по дате, дают возможность накапливать значения по документам за конкретный период (на каждый день, неделю, месяц и т. д.). В запросе может присутствовать несколько группировок такого типа.
Неделя	Week	
Месяц	Month	
Квартал	Quarter	
Год	Year	

Замечание: В программном модуле, где используется запрос, имя <ИмяГруппировки> (или <ПредопредГруппировка>) будет являться атрибутом запроса. Кроме того, это имя используется в качестве параметра метода запросов Группировка (см. главу «Работа с Запросами»).

Объекты, по которым запрос будет обрабатывать информацию и те значения которые он будет выбирать, упорядочивать и группировать во временном наборе данных, полностью определяются той внутренней переменной, на базе которой объявлена группировка (подробнее см. выше в разделе «Объявление внутренней переменной»).

Несколько операторов `Группировка`, следующих друг за другом в описании запроса, создают вложенные группировки запроса. Первая группировка — самая старшая, в нее будет вложена следующая группировка, далее будет вложена следующая и т. д. По смыслу, вложенная группировка осуществляет более детальный просмотр объекта внешней группировки. Например, если во внешней группировке мы просматриваем регистр, то во вложенной группировке можно просмотреть документы движений этого регистра, а далее можно сформировать группировку по номерам строк этих документов.

При описании вложенных группировок разработчик конфигурации должен ясно себе представлять суть и смысл образования вложенных группировок в формируемом запросе, поскольку интерпретация полученных результатов выполнения запроса полностью зависит от контекста решаемой задачи.

По умолчанию, документы упорядочиваются в группировке по дате и времени документов, элементы справочников — в зависимости от основного представления, заданного при описании справочника в конфигураторе (код или наименование). Однако критерий упорядочивания в группировке можно установить при помощи необязательного ключевого слова "`Упорядочить по`". Параметры, следующие за данным ключевым словом, определяют упорядочивание строк в группировке.

С параметром упорядочивания связан специальный метод для доступа к значениям объекта «Запрос». Речь идет об использовании метода `ЗначениеУпорядочивания`. В программном модуле, после того как запрос уже сформирован, мы можем при помощи этого метода получить значение поля упорядочивания из временного набора данных, не обращаясь к базе данных. Например, если у нас в запросе была группировка "`Товар`", а нам для формирования некоторого отчета нужны значения наименований товаров, то эти наименования товаров можно получить двумя способами:

```
Наим = Запрос.Товар.Наименование;
```

или

```
Наим = Запрос.ЗначениеУпорядочивания("Товар", 1);
```

Смысл использования данного специального метода доступа в том, что значения упорядочивания хранятся во временном наборе данных, сформированном запросом, поэтому за этими значениями программе нет необходимости снова обращаться к информационной базе, а можно получить непосредственно из временного набора. Эффект использования специального метода доступа может проявиться только в сетевой версии информационной базы при формировании очень больших отчетов, время формирования которых порядка десятков минут. В этом случае применение этого метода доступа даст некоторый выигрыш по времени.

Пример:

```
Процедура Группировки()
   Перем Запрос, ТекстЗапроса;
   Перем ДатаНач, ДатаКон;
   ЖР = СоздатьОбъект("ЖурналРасчетов.Зарплата");
   Нач = ЖР.НачалоТекущегоПериода();
   Кон = ЖР.КонецТекущегоПериода();
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса =
   "//{{ЗАПРОС(Группировки)
   |Период с ДатаНач по ДатаКон;
   |Рез = ЖурналРасчетов.Зарплата.Результат;
   |Расч = ЖурналРасчетов.Зарплата.ВидРасч;
   |Дни = ЖурналРасчетов.Зарплата.Дни;
   |Сотр = ЖурналРасчетов.Зарплата.Объект;
   |Группировка Сотр без групп;
   |Группировка Расч;
   |Функция Сум = Сумма(Рез);
   |Функция Дней = Сумма(Дни);
   |Условие(Рез о 0);
   |"//})ЗАПРОС
   ;
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
      Возврат;
   КонецЕсли;
   // Подготовка к заполнению выходных форм данными запроса
   Таб = СоздатьОбъект("Таблица");
   Пока Запрос.Группировка("Сотр") = 1 Цикл
      // Заполнение полей Сотр
      Таб.ВывестиСекцию("Сотр ");
      Пока Запрос.Группировка("Расч") = 1 Цикл
         // Заполнение полей
         Расч Таб.ВывестиСекцию("Расч");
      КонецЦикла;
```

```
   КонецЦикла;
   Таб.Опции(0, 0, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("Результат", );
   // Вывод заполненной формы
КонецПроцедуры
```

Без итогов

Не накапливать итоги по группировкам.

Синтаксис:

```
Без итогов;
```

Англоязычный синоним:

```
Without Totals;
```

Описание:

Цель данного оператора — увеличение скорости выполнения запроса, при условии, что итоговые записи при использовании запроса не нужны. Данный оператор языка запросов, дает возможность не накапливать итоги по группировкам. Использовать данный оператор в тексте запроса имеет смысл, когда запрос строится не для формирования сложного отчета, а например, для простой выборки значений из информационной базы для последующей их обработки.

В случае применения данного оператора в тексте запроса, при обходе результатов запроса применяется только один цикл обхода, используя метод объекта «Запрос» `Группировка()` без параметра.

Пример:

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса = "
...
|Группировка Товар;
|Группировка Склад;
|Без Итогов; ";
Если Запрос.Выполнить(ТекстЗапроса) = 1 Тогда
   Возврат;
КонецЕсли;

Пока Запрос.Группировка() = 1 Цикл
   ...
КонецЦикла;
```

Замечание. Если в тексте запроса используется группировка по многоуровневому справочнику и не указано "`Без Групп`", то итоги по группам справочника будут накапливаться. Другими словами, если в запросе не нужны итоги по группам справочника, то в тексте запроса кроме использования оператора "`Без Итогов`" дополнительно следует в операторах "`Группировка ...`" использовать ключевое слово "`Без Групп`".

Условие

Назначить условие включения информации в запрос.

Синтаксис:

```
Условие(<ЛогическоеВыражение>);
```

Англоязычный синоним:

```
Condition
```

Параметры:

`<ЛогическоеВыражение>` Логическое выражение встроенного языка 1С:Предприятие.

Описание:

Оператор `Условие` назначает условие включения информации в запрос. Если значение `<ЛогическоеВыражение>` верно, то информация включается в запрос, иначе нет.

Пример:

```
ТекстЗапроса =
"//{{ЗАПРОС(Одинарный)
|Период с ДатаНачала по ДатаКонец;
|Оклад = Справочник.Сотрудники.Оклад;
|Ктг = Справочник.Сотрудники.Категория;
|Группировка Ктг без групп;
|Функция Всего = Сумма(Оклад);
|Условие(Ктг.Выбран() = 1);   // только для тех сотрудников
| //у кого заполнен реквизит Категория
|"//}}ЗАПРОС
;
```

В логическом выражении могут участвовать как внутренние, так и внешние переменные запроса, т. е. переменные программного модуля, доступные в процедуре, использующей запрос.

Пример:

* Здесь показан отрывок текста процедуры, в которой формируется некоторый отчет, причем переменные:
ВыбТовар, ВыбОтдел, ВыбСотрудник
являются реквизитами диалога отчета, значит, они доступны в программном модуле, поэтому могут быть использованы в логическом выражении оператора Условие. В данном примере операторы Условие использованы для фильтрации в запросе только выбранных значений параметров отчета,

```
Запрос = СоздатьОбъект("Запрос");
ТекстЗапроса =
"//{{ЗАПРОС(Квоты)
|Товар = Регистр.КвотыТоваров.Товар;
|Отдел = Регистр.КвотыТоваров.Отдел;
|Сотрудник = Регистр.КвотыТоваров.Сотрудник;
|Партнер = Регистр.КвотыТоваров.Партнер;
|Количество = Регистр.КвотыТоваров.КвотаТовара;
|Группировка Товар;
|Группировка Отдел;
|Группировка Сотрудник;
|Группировка Партнер;
|Функция Кол_во = КонОст(Количество);
|Условие(Товар.ПринадлежитГруппе(ВыбТовар) = 1);
|Условие(Отдел = ВыбОтдел);
|Условие(Сотрудник = ВыбСотрудник.Сотрудник);
|"//}}ЗАПРОС
;
// Если ошибка в запросе, то выход из процедуры
Если Запрос.Выполнить(ТекстЗапроса)= 0 Тогда
   Возврат;
КонецЕсли;
```

В параметре оператора Условие, в принципе, можно записывать логические выражения любой сложности (по правилам встроенного языка 1С:Предприятие), используя любые логические операторы (>, <, =, >=, <>, И, ИЛИ, НЕ и другие), при этом в качестве операндов могут участвовать как внутренние, так и внешние переменные запроса (если существуют внутренняя и внешняя переменные с одинаковым именем, то использоваться по данному имени будет внутренняя переменная).

Пример:

```
Условие(Цена > 23.5);
Условие((Товар.ПринадлежитГруппе(ВыбТовар) = 1) И (Товар.Розн_Цена = 100));
```

Если в описании запроса использовано несколько операторов Условие, то они объединяются по логическому "И".

Пример:

* Использование двух операторов:

```
Условие(Товар.Наименование <> "Стол");
Условие(Товар.Наименование <> "Шкаф");
```

* аналогично записи одного оператора:

```
Условие((Товар.Наименование <> "Стол") И (Товар.Наименование<> "Шкаф"));
```

Логический оператор принадлежности

Кроме обычных логических операторов (>, <, =, >=, <=, <>, И, ИЛИ, НЕ) в операторе "Условие ..." языка запросов можно использовать дополнительный оператор — логический оператор принадлежности.

Синтаксис:

```
В
```

Англоязычный синоним:

```
In
```

Описание:

Оператор языка запросов "Условие (А в Б);" говорит о том, что условие истинно, когда значение А является подмножеством значения Б.

Следует особо отметить, что если значение Б пустое (объект не выбран), то условие является истинным, в отличии от оператора «=» (равно).

Если на принадлежность проверяется значение типа элемент справочника, то проверка выполняется с учетом его возможного вхождения в группу справочника. Аналогично, проверка на принадлежность субсчета осуществляется с учетом его возможного вхождения в счет-группу.

В качестве включающего подмножества логического оператора принадлежности (второй параметр после слова «в») может выступать как простое значение, так и список значений. В этом случае проверка выполняется с учетом вышеотмеченных особенностей для каждой строки списка значений.

Скорость выполнения запросов оптимизирована под использование оператора принадлежности, как в клиент-серверной, так и в файл-серверной версии системы 1С:Предприятие.

Внимание. Логический оператор принадлежности не поддерживается встроенным языком системы 1С:Предприятие, а применяется только в языке запросов.

Оператор принадлежности существенно облегчает написание текстов запроса, делает их более понятными.

Пример:

* без использования логического оператора принадлежности

```
ТекстЗапроса =
...
| Товар = Документ.ВидДокумента.Товар;";
...
Если ВыбТовар.Выбран() = 1 Тогда
   Если ВыбТовар.ЭтоГруппа() = 1 Тогда
      ТекстЗапроса = ТекстЗапроса +
               "Условие(Товар.ПринадлежитГруппе(ВыбТовар) = 1);";
   Иначе
      ТекстЗапроса = ТекстЗапроса + "Условие(Товар = ВыбТовар);";
   КонецЕсли
КонецЕсли;
```

* с использованием логического оператора принадлежности

```
ТекстЗапроса=
...
|Товар = Документ.ВидДокумента.Товар;";
...
|Условие (Товар в ВыбТовар);
...
```

Примеры использования Запросов

Печать каталога товаров

Далее приведена процедура, выполняющая печать всего справочника товаров с использованием формирования запроса. Для простого перебора справочника использование запросов оправдано только в том случае, если мы либо используем полученный временный набор данных многократно, или вычисляем функции, или производим нестандартное упорядочивание объектов. В данном примере запрос используется для сортировки справочника по некоторому реквизиту товара.

```
//-------------------------------
Процедура ПечатьСправочника()
   // Процедура печати полного справочника товаров
   Перем Запрос, ТекстЗапроса, Таб;
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(Печать)
   |Товар = Справочник.Товары.ТекущийЭлемент;
   |Группировка Товар Упорядочить по Товар.МинЗапас;
   |"//}}ЗАПРОС
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
      Возврат;
   КонецЕсли;
   // Заполнение выходных форм данными запроса
   // Создание Таблицы для выходного отчета
   Таб = СоздатьОбъект("Таблица");
   Таб.ВывестиСекцию("Отчет");
   Пока Запрос.Группировка("Товар") = 1 Цикл
      // Заполнение полей Товар
      Если Запрос.Товар.ЭтоГруппа() = 1 Тогда
         Таб.ВывестиСекцию("Группа");
      Иначе
         Таб.ВывестиСекцию("Товар");
      КонецЕсли;
   КонецЦикла;
   //Отображение выходного отчета
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 4, 0);
   Таб.Показать("Список товаров по каталогу", "");
КонецПроцедуры
```

Отчет по неходовым товарам

Далее приведен пример нетривиального использования запроса для просмотра одновременно многих видов документов. Цель данной процедуры — вывести в отчет перечень неходовых товаров, которые совсем не продавались за заданный период и показать в каждой строке текущий остаток и стоимость этих товаров. В данном примере запрос формируется с целью определить, что в него не вошло.

```
Процедура Сформировать()
```

```
    Перем Запрос, ТекстЗапроса, Таб;
    Рег = СоздатьОбъект("Регистр.ОстаткиТовара");
    Запрос = СоздатьОбъект("Запрос");
    ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
    |Период С ДатаНачала По ДатаКонца;
    |ТОВАР = Документ.РасходнаяБН.Товар,
    |   Документ.РасходнаяКредит.Товар, Документ.РасходнаяНал.Товар,
    |   Документ.РасходнаяРеализ.Товар, Документ.Счет.Товар;
    |Группировка ТОВАР упорядочить по ТОВАР.Наименование без групп;
    |"//}}ЗАПРОС
    ;
    Если ДатаКонца >= ПолучитьДатуТА() Тогда
       ТекстЗапроса = ТекстЗапроса + "Период С ДатаНачала;";
    Иначе
       ТекстЗапроса = ТекстЗапроса + "Период С ДатаНачала По ДатаКонца;";
       Рег.ВременныйРасчет();
       РассчитатьРегистрыНа(ДатаКонца);
    КонецЕсли;
    // Выполнение Запроса
    Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
    КонецЕсли;
    Тов = СоздатьОбъект("Справочник.Товары");
    // обход включая группировки
    Тов.ВключатьПодчиненные(1);
    // упорядочить по наименованиям
    Тов.ПорядокНаименозаний();
    ИтогоОстаток = 0;
    ИтогоСумма = 0;
    Таб = СоздатьОбъект("Таблица");
    Таб.ВывестиСекцию("Отчет");
    Состояние("В отчет выведено " + ЧислоСтрок + " строк.");
    // Запускаем полный цикл по товарам Справочника
    Тов.ВыбратьЭлементы();
    Пока Тов.ПолучитьЭлемент() > 0 Цикл
       Флаг = 0;
       Товар = Тов.ТекущийЭлемент();
       Если Товар.ЭтоГруппа() = 1 Тогда
          Продолжить;
       КонецЕсли;
       // Здесь пытаемся получить из Запроса информацию о товаре,
       // но используем просто сам факт того, что товар попал во
       // временный набор данных Запроса.
       // Если товар есть в Запросе, то значит он упоминался в
       // каких то документах,
       // иначе — товар не пользуется спросом — неходовой.
       Если Запрос.Получить(Товар) = 1 Тогда
          Продолжить;
       КонецЕсли;
       // находим остатки неходового товара на складе
       Рег.СводныеОстатки(Товар, );
       ТекОстаток = Рег.ОстатокТовара;
       ТекСумма = Рег.БазоваяСтоимость;
       Если ТекОстаток = 0 Тогда
          Продолжить;
       КонецЕсли;
       Таб.ВывестиСекцию("Товар");
       ИтогоОстаток = ИтогоОстаток + ТекОстаток;
       ИтогоСумма = ИтогоСумма + ТекСумма;
    КонецЦикла;
    Таб.ВывестиСекцию("Итоги");
    Таб.ТолькоПросмотр(1);
    Таб.Опции(0, 0, 3, 0);
    Таб.Показать("Отчет о неходовых товарах", "");
КонецПроцедуры

ДатаКонца = РабочаяДата();
ДатаНачала = ДатаКонца — Константа.ПериодАнализа;
```

Отчет по регистру с точностью до строки документа

Далее приведен пример использования запроса для просмотра регистра с выборкой документов, производивших движение данного регистра, с доступом к каждой строке этих документов. Цель данной процедуры — сформировать отчет, чтобы по каждому из видов топлива, отпущенных за заданный период покупателям, показать перечень номеров автомашин, которыми производилась отгрузка. Особенностью данного отчета является то, что номера автомашин не являются измерениями регистра, а зафиксированы в каждой строке отпускного документа.

Использование в описании внутренней переменной для регистра атрибута `НомерСтроки`, означает выборку связанных номеров строк тех документов, которые произвели движение по регистру (предполагается, что в конфигурации в Модулях документов перед движением данного регистра использовали метод `ПривязыватьСтроку`).

```
Процедура Сформировать()
   Перем Запрос, ТекстЗапроса, Таб;
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
   |Период с НачДата по КонДата;
   |ВидТоплива = Регистр.ПокупателиКолво.ВидыТоплива;
   |Вес = Регистр.ПокупателиКолво.Кг;
   |Покуп = Регистр.ПокупателиКолво.Покупатели;
   |Док = Регистр.ПокупателиКолво.ТекущийДокумент;
   |Ном = Регистр.ПокупателиКолво.НомерСтроки;
   |Группировка ВидТоплива;          //по измерению Регистра
   |Группировка Док;                 // по документам, двигавшим Регистр
   |Группировка Ном;                 //по номерам строк документов
   |Функция ВсегоКолво = КонОст(Вес);
   |Функция ПриходКолво = Приход(Вес);
   |Условие(Покуп = ВыбПокупатель);
   |"//}}ЗАПРОС
   ;
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
      Возврат;
   КонецЕсли;
   // Заполнение выходных форм данными Запроса
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("Таблица1");
   Таб.ВывестиСекцию("Шапка");
   Пока Запрос.Группировка(1) = 1 Цикл
      // Отображаем значение измерения — ВидТоплива
      Таб.ВывестиСекцию("ВидТоплива");
      // запускаем вложенную группировку по документам,
      // которые производили движения Регистра
      Пока Запрос.Группировка(2) = 1 Цикл
         Док1 = Запрос.Док;
         // отфильтруем нужные нам документы
         Если НЕ(Док1.Вид() = "Продажа") Тогда
            Продолжить;
         КонецЕсли;
         // запускаем вложенную группировку по строкам
         // документа, связанным с движениями регистра
         Пока Запрос.Группировка(3) = 1 Цикл
            // Получаем в документе строку по найденному номеру
            Док1.ПолучитьСтрокуПоНомеру(Запрос.Ном);
            Таб.ВывестиСекцию("Строка");
         КонецЦикла;
      КонецЦикла;
   КонецЦикла;
   // Вывод заполненной формы
   Таб.Опции(О, О, 0, 0);
   Таб.ТолькоПросмотр(1);
   Таб.Показать("", "");
КонецПроцедуры
```

Анализ счета

Приводим пример использования запроса для работы с бухгалтерскими операциями и проводками. Запрос обрабатывает корреспонденции счета `Сч` по счетам `КорСч` за расчетный период. Значение счета для анализа задается в диалоге установкой `ВыбСч`.

```
Процедура АнализСчета()
```

```
    Перем Запрос, ТекстЗапроса, Таб;
    //Создание объекта типа Запрос
    Запрос = СоздатьОбъект("Запрос");
    ТекстЗапроса = "//{{ЗАПРОС(Сформировать)
    |Период с ДатаС по ДатаПо;
    |Сч = Операция.Счет;
    |КорСч = Операция.КорСчет;
    |Сумма = Операция.Сумма;
    |Группировка Сч упорядочить по Сч.Код;
    |Группировка КорСч упорядочить по КорСч.Код;
    |Функция КорДо = КорДО(Сумма);
    |Функция КорКо = КорКО(Сумма);
    |Условие(Сч = ВыбСч);
    |"//}}ЗАПРОС
    ;
    // Если ошибка в запросе, то выход из процедуры
    Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
    КонецЕсли;
    // Подготовка к заполнению выходных форм данными запроса
    Таб = СоздатьОбъект("Таблица");
    Таб.ИсходнаяТаблица("Сформировать");
    // Заполнение полей "Заголовок"
    Таб.ВывестиСекцию("Заголовок");
    Состояние("Заполнение выходной таблицы...");
    Пока Запрос.Группировка("Сч") = 1 Цикл
       // Заполнение полей Сч
       Таб.ВывестиСекцию("Сч");
       Пока Запрос.Группировка("КорСч") = 1 Цикл
          // Заполнение полей КорСч
          Таб.ВывестиСекцию("КорСч");
       КонецЦикла;
    КонецЦикла;
    // Заполнение полей "Итого"
    Таб.ВывестиСекцию("Итого");
    // Вывод заполненной формы
    Таб.Опции(1, 0. 1, 0);
    Таб.Показать("Сформировать", "");
КонецПроцедуры
```

Разработка вложенных отчетов

Средства программы 1С:Предприятие по работе с Таблицами позволяют создавать эффектные отчеты, причем на экране монитора отображается не просто мертвая картинка предварительного просмотра печати, а живой отчет, который можно редактировать или непосредственно из него вызывать дополнительную поясняющую информацию.

Поскольку каждая ячейка Таблицы может содержать значение, записанное в нее (см. Конфигуратор, редактор таблиц — Свойства ячейки — Текст — поле: Значение), то в программном модуле формы отчета возможно обрабатывать это значение. Обработка значения ячейки Таблицы вызывается системой по клавише <Enter> или по двойному щелчку мышью на какой-либо ячейке (если режим «только просмотр»). Стандартными действиями системы на обработку такого события являются: для документа — открытие документа, для элемента справочника — открытие формы редактирования элемента справочника. Другими словами, стандартные действия системы зависят от типа данных содержащегося в ячейке значения. Однако, это событие возможно перехватить и обработать нестандарным способом. Для этого предназначена предопределенная процедура встроенного языка `ОбработкаЯчейкиТаблицы`.

Примером нестандартной обработки значения ячейки таблицы может быть, например, вызов на формирование другого отчета. Таким образом, мы можем создавать как бы вложенные отчеты, которые вызываются один из другого, выдавая с каждым разом более детальную информацию. Допустим отчет «Взаиморасчеты» при формировании всегда выводится в кратком виде, когда виден только сводный баланс по контрагенту. Для того, чтобы получить детальный отчет по данному контрагенту, достаточно встать курсором в готовой форме отчета на этого контрагента и нажать клавишу <Enter>. Тогда сработает процедура `ОбработкаЯчейкиТаблицы`, в которой можно записать вызов формирования детального отчета. А если, кроме того, завести флаг режима отображения, то можно вместо этого показывать карточку этого контрагента из справочника.

Рассмотрим построение вложенных отчетов на примере. Допустим, у вас есть отчет «ПродажиТоваров», в котором отображается перечень товаров, количество и сумма проданных за некоторый период товаров. Программный модуль формирования такого отчета приведен ниже.

Пример:

```
Процедура ПродВсего()
    Перем Запрос, ТекстЗапроса, Таб;
    ДатаКон = ДатаКонца;
    Если ДатаКон >= ПолучитьДатуТА() Тогда
```

```
      ДатаКон = Дата(0);
   КонецЕсли;
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(ПродВсего)
   |Период с ДатаНачала по ДатаКон;
   |ТОВАР = Документ.РасхНакл.Товар;
   |Сумма_Прод = Документ.РасхНакл.СуммаРуб;
   |КОЛВО_Прод = Документ.РасхНакл.Количество;
   |Группировка ТОВАР;
   |Функция Продано = Сумма(КОЛВО_Прод);
   |Функция СуммаПродано = Сумма(Сумма_Прод);
   |"//}}ЗАПРОС
   ;
   // Если ошибка в запросе, то выход из процедуры
   Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
      Возврат;
   КонецЕсли;
   // Подготовка к заполнению
   Таб = СоздатьОбъект("Таблица");
   Таб.ИсходнаяТаблица("ТабВсего");
   Таб.ВывестиСекцию("Отчет");
   Пока Запрос.Группировка("Товар") = 1 Цикл
      ПродСумма = Запрос.СуммаПродано;
      Если Запрос.Товар.ЭтоГруппа() = 1 Тогда
         Таб.ВывестиСекцию("Группа");
      Иначе
         Таб.ВывестиСекцию("Товар");
      КонецЕсли;
   КонецЦикла;
   // Вывод заполненной формы
   Таб.ТолькоПросмотр(1);
   Таб.Опции(0, 0, 4, 0);
   Таб.Показать("Продажа товаров ", "");
КонецПроцедуры
```

В режие исполнения у нас получится такой отчет:

Отчет по продаже товаров
с 01.11.96 по 19.11.96

Товар	Продано	
	Кол-во	Сумма
Обувь	12	1270
Сапоги женские	5	920
Туфли мужские	7	350

Далее, допустим, мы хотим получить более подробный отчет по конкретному товару, так, чтобы указав на товар в отчете, и дважды нажав кнопку мыши, мы получали отчет по продажам именно этого товара с точностью до каждого клиента. Для этого откроем в Конфигураторе редактор таблицы нашего отчета и в свойствах ячейки (Свойства ячейки — Текст — поле: Значение), отображающей наименование товара, проставим значение ячейки. («Запрос.Товар»)

Далее, в программном модуле напишем процедуру формирования дополнительного отчета.

Продолжение примера:

```
//*********************************
// Процедура дополнительного отчета
Процедура ПродТовар(ВТовар)
   Перем Запрос, ТекстЗапроса, Таб;
   ДатаКон = ДатаКонца;
   Если ДатаКон >= ПолучитьДатуТА() Тогда
      ДатаКон = Дата(0);
   КонецЕсли;
   //Создание объекта типа Запрос
   Запрос = СоздатьОбъект("Запрос");
   ТекстЗапроса = "//{{ЗАПРОС(ПродТовар)
   |Период с ДатаНачала по ДатаКон;
   |КЛИЕНТ = Документ.РасхНакл.Клиент;
   |ТОВАР = Документ.РасхНакл.Товар;
   |СУММ = Документ.РасхНакл.СуммаВал;
   |КОЛВО = Документ.РасхНакл.Количество;
```

```
    |Группировка КЛИЕНТ Упорядочить По КЛИЕНТ.Наименование;
    |Группировка ТОВАР Упорядочить По ТОВАР.Наименование;
    |Функция Продано = Сумма(КОЛВО);
    |Функция ПродСум = Сумма(СУММ);
    |"//}}ЗАПРОС
    ;
    Если ВТовар.Выбран() = 1 Тогда
       Если ВТовар.ЭтоГруппа() = 1 Тогда
          ТекстЗапроса = ТекстЗапроса +
              "Условие(Товар.ПринадлежитГруппе(ВТовар) = 1);";
       Иначе
          ТекстЗапроса = ТекстЗапроса + "Условие (Товар = ВТовар);";
          ФОдинТовар = 1;
       КонецЕсли;
    КонецЕсли;
    // Если ошибка в запросе, то выход из процедуры
    Если Запрос.Выполнить(ТекстЗапроса) = 0 Тогда
       Возврат;
    КонецЕсли;
    // Подготовка к заполнению выходных форм
    Таб = СоздатьОбъект("Таблица");
    Таб.ВывестиСекцию("Отчет");
    Пока Запрос.Группировка("Клиент") = 1 Цикл
       Таб.ВывестиСекцию("Клиент");
       Пока Запрос.Группировка("Товар") = 1 Цикл
          Если ФОдинТовар = 0 Тогда
             Таб.ВывестиСекцию("Товар");
          КонецЕсли;
       КонецЦикла;
    КонецЦикла;
    // Вывод заполненной формы
    Таб.ТолькоПросмотр(1);
    Таб.Опции(0, 4, 0, 0);
    Таб.Показать("Продажа товара", "");
КонецПроцедуры
```

Теперь нам осталось написать предопределенную процедуру, которая возьмет на себя обработку события «двойной щелчок мышью на выбранной ячейке таблицы». Главная задача этой процедуры — определить, что выбрана ячейка, где указано значение товара и вызвать на исполнение процедуру формирования дополнительного отчета, написанную ранее.

Продолжение примера:

```
//-------------------------
Процедура ОбработкаЯчейкиТаблицы(ЗначЯч, ФлагСтандартнойОбработки)
   Если ТипЗначения(ЗначЯч) = 2 Тогда
      ФлагСтандартнойОбработки = 1;
      Возврат;
   КонецЕсли;
   ПродТовар(ЗначЯч);
КонецПроцедуры
```

Теперь, в режиме исполнения, если в отчете «Продажи товаров» мы укажем курсором на товар, например, «Сапоги женские» и дважды щелкнем на нем мышью, то сформируется дополнительный подробный отчет «Продажа товара».

Способы оптимизации формирования отчетов

Процесс получения отчетов с использованием запросов можно условно разделить на две фазы: сначала формируется запрос, затем полученные данные выводятся в Таблицу. В данном разделе речь пойдет о второй фазе получения отчета — выводе данных в Таблицу.

Рассмотрим следующий пример. Допустим, требуется вывести в отчет полный перечень товаров со всеми их реквизитами. Для этого сначала формируется запрос с одной Группировкой — "Товар", затем полученные данные выводятся в Таблицу. Ниже приведены три варианта вывода данных в Таблицу.

Вариант 1

В программном модуле текст вывода данных в Таблицу следующий

```
Пока Запрос.Группировка("Товар") = 1 Цикл
   Таб.ВывестиСекцию("Товар");
КонецЦикла;
```

Секция "Товар" в Таблице имеет следующий формат:

<Запрос.Товар.Наименование>	<Запрос.Товар.Артикул>
<Запрос.Товар.Цена>	<Запрос.Товар.Валюта>

В первом варианте отображаемые реквизиты товара полностью вычисляются в ячейках Таблицы, причем доступ к каждому реквизиту товара происходит по полному пути: Запрос-Товар-Реквизит. Данный вариант вывода данных в Таблицу самый медленный.

Вариант 2

В программном модуле текст вывода данных в Таблицу следующий

```
Пока Запрос.Группировка("Товар") = 1 Цикл
   ПечНаим = Запрос.Товар.Наименование;
   ПечАртикул = Запрос.Товар.Артикул;
   ПечЦена = Запрос.Товар.Цена;
   ПечВалюта = Запрос.Товар.Валюта;
   Таб.ВывестиСекцию("Товар1");
КонецЦикла;
```

Секция "Товар1" в Таблице имеет следующий формат:

<ПечНаим>	<ПечАртикул>
<ПсчЦена>	<ПечВалюта>

Во втором варианте отображаемые реквизиты товара вычисляются в программном модуле, а в ячейках Таблицы размещены простые выражения — ссылки на идентификаторы программного модуля. Доступ к каждому реквизиту товара происходит по полному пути: Запрос-Товар-Реквизит. В данном варианте фаза вывода данных в Таблицу работает быстрее первого варианта (выигрыш ~20%), т. к. в программном модуле выражения вычисляются существенно быстрее, чем выражения, помещенные в ячейках Таблицы (программный модуль при загрузке компилируется, а выражения в ячейках таблицы интерпретируются каждый раз при выводе секций).

Вариант 3

В программном модуле текст вывода данных в Таблицу следующий

```
Пока Запрос.Группировка("Товар") = 1 Цикл
   Тов = Запрос.Товар;
   ПечНаим = Тов.Наименование;
   ПечАртикул = Тов.Артикул;
   ПечЦена = Тов.Цена;
   ПечВалюта = Тов.Валюта;
   Таб.ВывестиСекцию("Товар1");
КонецЦикла;
```

Секция "Товар1" в Таблице имеет следующий формат:

<ПечНаим>	<ПечАртикул>
<ПсчЦена>	<ПечВалюта>

В третьем варианте отображаемые реквизиты товара вычисляются в программном модуле, а в ячейках Таблицы размещены простые выражения — ссылки на идентификаторы программного модуля. Доступ к каждому реквизиту товара происходит по сокращенному пути — через промежуточную переменную: Товар — Реквизит. В данном варианте фаза вывода данных в Таблицу работает быстрее, чем во втором варианте (выигрыш ~20%) и существено быстрее, чем в первом варианте (выигрыш ~40%), т. к. вычисление значений реквизитов объектов через «одну точку» выполняется быстрее, чем через «две (и более) точки».

Кратко суть этого раздела можно выразить так: при реализации больших отчетов, которые рассчитаны на отображение более сотни строк, следует придерживаться следующих правил:

- не размещайте сложных выражений в ячейках Таблицы. Лучше вычислить необходимые значения непосредственно в программном модуле;
- если необходимо получить несколько реквизитов одного объекта, который сам является составной частью другого объекта, то следует воспользоваться промежуточной переменной, в которую сначала можно записать значение всего объекта «целиком», а затем уже от нее получать требуемые данные.

Глава 35
Работа с Картинками

При формировании пользовательского интерфейса прикладной задачи иногда требуется отображать в диалоговых формах и печатных документах графические файлы. Для этого в системе 1С:Предприятие используется специальный агрегатный тип данных — *Картинка* и специальнй элемент формы диалога и таблицы: «Картинка».

Одним из примеров использования данного объекта может служить диалоговая форма справочника «Сотрудники» в которой при выборе сотрудника отображается его фотография.

Контекст работы с картинками

Если в форму диалога или в таблицу при помощи визуальных средств конфигуратора вставлены элементы формы типа «Картинка», система автоматически создает объекты этого типа, причем идентификаторы этих элементов доступны в контексте программного модуля этой формы как уже существующие объекты типа «Картинка». Кроме того, в программных модулях допускается создавать произвольное число объектов типа «Картинка» просто при помощи вызова системной функции `СоздатьОбъект`.

При создании объекта типа «Картинка» при помощи функции `СоздатьОбъект`, в качестве названия агрегатного типа данных обязательно должно выступать ключевое слово `Картинка`.

Англоязычный синоним ключевого слова `Картинка` — `Picture`.

Пример:

```
Фото=СоздатьОбъект("Картинка");
```

Методы объекта Картинка

Загрузить

Загрузить из файла.

Синтаксис:

`Загрузить(<ИмяФайла>)`

Англоязычный синоним:

`Load`

Параметры:

`<ИмяФайла>`	Строковое выражение, которое может задавать или идентификатор картинки в библиотеке картинок конфигурации или имя файла картинки.

Описание:

Метод `Загрузить` загружает картинку из файла с указанным именем `<ИмяФайла>` или берет по идентификатору картинку из библиотеки картинок конфигураци. Файл может быть следующих форматов:

.wmf .emf .ico .bmp .dib .rle

Пример:

```
Фото = СоздатьОбъект("Картинка");
Фото.Загрузить("c:\BMP\Boris.bmp");
```

Сохранить

Сохранить в файл.

Синтаксис:

`Сохранить(<ИмяФайла>)`

Англоязычный синоним:

`Save`

Параметры:

`<ИмяФайла>`	Строковое выражение — имя файла.

Описание:

Метод `Сохранить` выгружает картинку в файл с именем `<ИмяФайла>` в формате загрузки (загрузили в формате bmp — в этом формате и сохраним).

Пример:

```
Фото.Загрузить("c:\BMP\Boris.bmp");
Фото.Сохранить("c:\1Cv7\DBmy\Boris.bmp");
```

РежимРисования

Установить режим рисования картинки.

Синтаксис:

`РежимРисования(<Режим>)`

Англоязычный синоним:

`SetDrawMode`

Параметры:

`<Режим>`	Числовое выражение: 1 — растягивать; 2 — рисовать по центру с оригинальным размером; 3 — рисовать сохраняя оригинальное соотношение высоты и ширины.

Возвращаемое значение:

Текущее числовое значение режима рисования картинки (на момент до исполнения метода).

Описание:

Метод РежимРисования устанавливает режим рисования картинки.

Пример:

```
Фото.РежимРисования(2);
Фото.Загрузить("c:\1Cv7\DBmy\Boris.bmp");
```

УстановитьКартинку

Переустановить объект типа "Картинка".

Синтаксис:

```
УстановитьКартинку(<Объект>)
```

Англоязычный синоним:

```
SetPicture
```

Параметры:

<Объект>	Выражение, имеющее значение оъекта типа «Картинка» встроенного языка или OLE-объект типа I Picture.

Описание:

Метод УстановитьКартинку переписывает (копирует) значение источника <Объект> в текущий объект. (Данный метод копирует все содержимое одного объекта в другой, в отличие от оператора присваивания (=), который передает значение ссылки на уже существующий объект). Например:

```
A = СоздатьОбъект("Картинка");
B = A;
// в этом случае переменные A и B ссылаются на один и тот же объект
```

Рассмотрим другой вариант:

```
A = СоздатьОбъект("Картинка");
B = СоздатьОбъект("Картинка");
A.Загрузить("c:\BMP\Boris.bmp");
B.УстановитьКартинку(A);
// в этом случае переменные A и B ссылаются на два
// разных объекта, хотя и содержат одно и то же
```

Пример:

```
// Допустим в Форме диалога у нас есть элемент типа картинка
// с идентификатором Кино.
// Сделаем анимацию изображения в этой форме
Перем фото[5];
// создадим объекты и загрузим в них картинки
Для Н = 1 По 5 Цикл
   Фото[Н] = СоздатьОбъект("Картинка");
   Фото[Н].Загрузить("c:\BMP\Foto" + Строка(Н)+ ".bmp");
КонецЦикла;
// Теперь запустим анимацию
Для Раз = 1 По 100 Цикл
   Для Н = 1 По 5 Цикл
      Кино.УстановитьКартинку(Фото[Н]);
   КонецЦикла;
КонецЦикла;
```

Глава 36
Работа с Диаграммами

В настоящий момент существует широкий ряд программных продуктов, позволяющих, в дополнение к табличным отчетам, получать графические отчеты (диаграммы). Одними из наиболее известных продуктов, выполняющих эти функции, являются MS Graph97 и MS Excel chart, система деловой графики, входящая в комплект Lotus 1-2-3 фирмы «Lotus», VCI First impression chart от Visual Components и многие другие.

Все перечисленные продукты могут использоваться (через механизм OLE Automation) для построения диаграмм в системе 1С:Предприятие. Однако 1С:Предприятие имеет собственный встроенный объект, который, из всего многообразия возможностей продуктов данного класса, выполняет наиболее актуальные функции, необходимые для задач, решаемых системой программ 1С:Предприятие.

Настоящая глава является руководством по использованию объекта «Диаграмма» системы 1С:Предприятия.

Использование данной возможности имеет следующие преимущества:

- не требуется наличия на компьютере программных продуктов других производителей;
- за счет более тесного взаимодействия с системой 1С:Предприятие, объект «Диаграмма» обеспечивает большую скорость построения диаграмм;
- интеграция с 1С:Предприятием предоставляет дополнительные возможности (расшифровка диаграммы);
- простота использования — система атрибутов и методов достаточно проста, и ее освоение не потребует больших усилий.

Объект «Диаграмма» предназначен для применения только в табличных документах системы 1С:Предприятие и не может использоваться самостоятельно.

Контекст работы с диаграммами

Для использования диаграмм нужно в табличный документ 1С:Предприятия внедрить объект «Диаграмма», используя специальную кнопку инструментальной панели или через главное меню <Таблица>-<Вставить рисунок>-<Диаграмма>.

Процесс внедрения объекта «Диаграмма» в табличный документ подробно описан в документации к 1С:Предприятию (см. книгу «Руководство по конфигурированию и администрированию»). После внедрения объект нужно активизировать, например, двойным щелчком указателя мыши, и настроить его внешний вид.

Настройка диаграммы обычно проводится на этапе конфигурирования. Задача настройки состоит в определении формата, специфического для каждой из областей диаграммы, их размера и расположения.

В свойствах объекта есть закладка «Текст». В поле «Текст» этой закладки нужно внести вызов процедуры, управляющей объектом диаграммы. Сам объект «Диаграмма», в момент вызова этой процедуры, является *текущим объектом* таблицы (см. атрибут объекта «Таблицы» `ТекущийОбъект`) и передается в процедуру как параметр. Например, этот вызов будет выглядеть так:

```
ПостроитьДиаграмму(Таб.ТекущийОбъект);
```

Наполнение диаграммы данными осуществляется в теле вызываемой процедуры с помощью атрибутов и методов данного объекта.

Таким образом, во всех программных модулях доступ к атрибутам и вызов методов деловой графики может выполняться только при помощи переменной со ссылкой на объект типа «Диаграмма». Данный объект внедряется в таблицу 1С:Предприятие при конфигурировании. При инициировании построения диаграммы, значение этого объекта в качестве фактического параметра передается в обрабатывающую процедуру, где ссылка на объект становится доступна через идентификатор фиктивного параметра. Чтобы вызвать метод деловой графики, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

См. также: `ТекущийОбъект`

Объект «Диаграмма» — основные принципы и понятия, используемые при визуальной настройке и управлении

Диаграмма представляет собой определенную фигуру, которая строится на основе матрицы данных размером [M, N], где N — количество серий значений (далее серий), M — количество точек, замерами в которых получены значения серий.

Для количественной ориентации в диаграмме используются координатные оси. Каждому значению, отмеченному на оси, соответствует поясняющее имя (метка). В зависимости от типа координатной оси, метка может быть рассчитанной, или определяться при заполнении данными.

С точки зрения настройки, диаграмма состоит из четырех областей:

- область диаграммы — совокупность всех областей образующих диаграмму;
- область построения диаграммы — содержит координатное пространство и изображенную на нем фигуру;
- заголовок диаграммы;
- легенда — вспомогательная область, содержащая список меток, соответствующих сериям.

Настройка размеров и положения областей диаграммы производится визуально. Подробнее о порядке визуальной настройки формата диаграммы следует читать в книге «Руководство по конфигурированию и администрированию».

Атрибуты диаграммы

Заголовок

Заголовок диаграммы.
Синтаксис:
`Заголовок`
Англоязычный синоним:
`TitleText`
Описание:
Атрибут `Заголовок` содержит текст заголовка диаграммы. По умолчанию имеет значение «Диаграмма».
Пример:
```
Диаграмма.Заголовок = "Пример использования диаграммы";
```

Методы диаграммы

КоличествоСерий

Явно устанавливает количество серий диаграммы.
Синтаксис:
`КоличествоСерий(<Количество>)`
Англоязычный синоним:
`SeriesCount`
Параметры:

`<Количество>`	Числовое выражение, которое задает количество серий диаграммы.

Возвращаемое значение:
Текущее значение количества серий (до исполнения метода).
Описание:
Метод `КоличествоСерий` явно устанавливает количество серий диаграммы. Количество серий может быть изменено и в режиме визуальной настройки. Альтернативой использования этого метода является неявное изменение количества серий: в случае если в метод, одним из параметров, использующий номер серии, передается значение превышающее количество серий, количество серий будет неявно увеличено.
Пример:
```
Диагр.КоличествоСерий(3);        // установим количество серий 3
```

КоличествоТочек

Явно устанавливает количество точек диаграммы.
Синтаксис:
`КоличествоТочек(<Количество>)`
Англоязычный синоним:
`PointsCount`
Параметры:

`<Количество>`	Числовое выражение, которое задает количество точек диаграммы.

Возвращаемое значение:
Текущее значение количества точек (до исполнения метода).
Описание:
Метод `КоличествоТочек` явно устанавливает количество точек диаграммы. Количество точек может быть изменено и в режиме визуальной настройки. Альтернативой использования этого метода является неявное изменение количества точек: в случае если в метод, одним из параметров, использующий номер точек, передается значение превышающее количество точек, количество точек будет неявно увеличено.
Пример:
```
Диагр.КоличествоТочек(4);       // установим количество точек 4
```

УстановитьИмяСерии

Устанавливает имя серии.
Синтаксис:
`УстановитьИмяСерии(<НомерСерии>, <Имя>)`
Англоязычный синоним:
`SetSeriesLabel`
Параметры:

`<НомерСерии>`	Числовое выражение, которое задает номер серии, для которой устанавливается имя.
`<Имя>`	Строка, используемая для обозначения серии в легенде и подписях к координатным осям.

Описание:
Метод `УстановитьИмяСерии` явно устанавливает имя для заданного номера серии.
Пример:
```
Диаграмма.УстановитьИмяСерии(СчетчикСерий, Запрос.Товар.Наименование);
```

УстановитьИмяТочки

Устанавливает имя точки.

Синтаксис:

`УстановитьИмяТочки(<НомерТочки>, <Имя>)`

Англоязычный синоним:

`SetPointLabel`

Параметры:

`<НомерТочки>`	Числовое выражение, которое задает номер точки, для которой устанавливается имя.
`<Имя>`	Строка, используемая для обозначения точки в подписях к координатным осям.

Описание:

Метод `УстановитьИмяТочки` явно устанавливает имя для заданного номера точки.

Пример:

```
Диаграмма.УстановитьИмяТочки(1, "Продано на сумму");
```

ЦветСерии

Устанавливает цвет серии.

Синтаксис:

`ЦветСерии(<НомерСерии>, <Красный>, <Зеленый>, <Синий>)`

Англоязычный синоним:

`SetSeriesColor`

Параметры:

`<НомерСерии>`	Числовое выражение, которое задает номер серии, для которой устанавливается цвет.
`<Красный>`	Число, задающее красную компоненту цвета.
`<Зеленый>`	Число, задающее зеленую компоненту цвета.
`<Синий>`	Число, задающее синюю компоненту цвета.

Описание:

Метод `ЦветСерии` устанавливает RGB цвет для серии `<НомерСерии>`. По умолчанию первые 54 серии имеют уникальный цвет. Серии с большими номерами изображаются в диаграмме повторяя уже использованный цвет, но перестают окрашиваться сплошным цветом, используя более сложный способ заливки.

Пример:

```
Диаграмма.ЦветСерии(1, 35, 67, 89);
```

АвтоУстановкаИменСерий

Автоматическая установка имен серий.

Синтаксис:

`АвтоУстановкаИменСерий(<флаг>)`

Англоязычный синоним:

`AutoSeriesLabels`

Параметры:

`<Флаг>`	Числовое значение: 1 — разрешающий. 0 — запрещающий автогенерацию имен серий.

Описание:

Метод `АвтоУстановкаИменСерий` устанавливает режим автогенерации имен серий диаграммы. В случае, если автогенерация имен разрешена, сериям диаграммы будут автоматически присваиваться имена Серия1, Серия2, ..., СерияN. По умолчанию автогенерация запрещена.

Пример:

```
Диаграмма.АвтоУстановкаИменСерий(1);
```

АвтоУстановкаИменТочек

Автоматическая установка имен точек.

Синтаксис:

`АвтоУстановкаИменТочек(<флаг>)`

Англоязычный синоним:

`AutoPointLabels`

Параметры:

`<Флаг>`	Числовое значение: 1 — разрешить автогенерацию имен точек. 0 — запретить автогенерацию имен точек.

Описание:

Метод `АвтоУстановкаМменТочек` устанавливает режим автогенерации имен точек диаграммы. В случае, если автогенерация имен разрешена, точкам диаграммы будут автоматически присваиваться имена 1, 2, ..., N. По умолчанию автогенерация запрещена.

Пример:

```
Диаграмма.АвтоУстановкаИменТочек(1);
```

УстановитьЗначение

Установка значения в заданной точке и серии.

Синтаксис:
УстановитьЗначение(<НомерТочки>, <НомерСерии>, <Значен>, <Расшифровка>)
Англоязычный синоним:
SetValue
Параметры:

<НомерТочки>	Числовое выражение, которое задает номер точки, для которой устанавливается значение.
<НомерСерии>	Числовое выражение, которое задает номер серии, для которой устанавливается значение.
<Значен>	Числовое значение — задаваемое значение диаграммы.
<Расшифровка>	Необязательный параметр. Значение любого типа, используемое для расшифровки диаграммы при помощи предопределенной процедуры ОбработкаЯчейкиТаблицы.

Описание:
Метод УстановитьЗначение устанавливает значение серии <НомерСерии> в точке <НомерТочки> равным <Значен> с возможностью его последующей расшифровки.
Расшифровка может быть реализована при помощи предопределенной процедуры ОбработкаЯчейкиТаблицы. Вызов предопределенной процедуры ОбработкаЯчейкиТаблицы на исполнение производится в системе 1С:Предприятие по двойному щелчку мыши в табличном документе на выбранной точке диаграммы. При этом значение расшифровки <Расшифровка> этой точки диаграммы автоматически передается в качестве первого параметра в вызове предопределенной процедуры.
Пример:
```
Диаграмма.УстановитьЗначение(1, 5, 14);
```
См. также: ОбработкаЯчейкиТаблицы

Обновление

Установка режима обновления диаграммы.
Синтаксис:
Обновление(<флаг>)
Англоязычный синоним:
RedrawFlag
Параметры:

<Флаг>	Числовое значение: 1 — разрешить перерисовку диаграммы. 0 — запретить перерисовку диаграммы.

Описание:
Метод Обновление устанавливает режим обновления диаграммы. Запрет перерисовки диаграммы рекомендуется в целях ускорения работы системы на ьремя наполнения данными. По умолчанию перерисовка разрешена.
Пример:
```
Диаграмма.Обновление(1);          // включим перерисовку
```

Очистить

Очищает диаграмму.
Синтаксис:
Очистить()
Англоязычный синоним:
Clear
Описание:
Метод Очистить очищает диаграмму, т. е. удаляет все ранее установленные значения; количество серий и точек устанавливает нулевым; стирает все ранее установленные имена.
Пример:
```
Диагр.Очистить();       // очистить диаграмму
```

Пример использования

Предположим, что проектируется отчет о продаже товара. Внешний вид отчета визуально настроен. Мы предполагаем, что будет построена объемная гистограмма с одной точкой («Продано на сумму») и количеством серий равным количеству товаров.
Фрагментом модуля отчета, является процедура ПостроитьДиаграмму. Вызов этой процедуры находится в закладке «Текст» свойств внедренного в табличный документ, объекта «Диаграмма», он выглядит так:
```
ПостроитьДиаграмму(Таб.ТекущийОбъект, Запрос)
```
Параметрами процедуры являются:

- Объект «Диаграмма», переданный как текущий объект табличного документа (подробнее см. в главе Работа с таблицами);
- запрос к базе данных, сформированный и выполненный ранее.

Пример:
```
Процедура ПостроитьДиаграмму(Диаграмма, Запрос)
   //На время наполнения данными, запретим перерисовку
   Диаграмма.Обновление(0);
   // задаем текст заголовка
   Диаграмма.Заголовок = "Пример использования диаграммы";
```

```
    // задаем текст метки точки
    Диаграмма.УстановитьИмяТочки(1, "Продано на сумму");
    СчетчикСерий = 1;
    Пока Запрос.Группировка("Товар") = 1 Цикл
        ПродСумма = Запрос.СуммаПродано;
        // задаем очередное значение
        Диаграмма.Значение(1, СчетчикСерий, ПродСумма);
        // задаем метку очередной серии
        Диаграмма.УстановитьИмяСерии(СчетчикСерий, Запрос.Товар.Наименование)
        СчетчикСерий = СчетчикСерий + 1;
    КонецЦикла;
    // После заполнения данных, включим перерисовку
    Диаграмма.Обновление(1);
КонецПроцедуры
```

Глава 37
Работа с Файлами

Для работы с файлами в системе используется специальный агрегатный тип данных — ФС. Механизм работы с файлами предназначен для обеспечения возможности манипулирования файлами непосредственно из встроенного языка программы 1С:Предприятия.

Контекст работы с Файлами

Во всех программных модулях для работы с файлами следует использовать методы объекта типа «ФС». По умолчанию в системе всегда доступен уже существующий объект с именем ФС (англоязычный синоним — FS) (имя существующего объекта совпадает с названием типа данных), к которому можно непосредственно применять методы объекта ФС. Кроме того, можно создать произвольное число объектов типа «ФС» при помощи функции СоздатьОбъект. Чтобы вызвать метод объекта, имя метода (с указанием необходимых параметров) пишется через точку после имени объекта.

Англоязычный синоним ключевого слова ФС — FS.

Пример:

* Здесь приведены примеры различных способов вызова методов объекта «ФС».

```
ФС.УдалитьФайл("Текст1.txt");
ВАКФайлы = СоздатьОбъект("ФС");
ВАКФайлы.УдалитьФайл("Текст3.txt");
```

Методы объекта «ФС»

ВыбратьФайл

Открывает окно диалога выбора/сохранения файла.

Синтаксис:

```
ВыбратьФайл(<ТипДиалога>, <ИмяФайла>, <ИмяНачКаталога>, <ЗаголовокОкна>, <Фильтр>,
            <Расширение>, <Таймаут>)
```

Англоязычный синоним:

```
SelectFile
```

Параметры:

<ТипДиалога>	Числовое выражение, значение которого определяет тип открываемого диалога. Допустимые значения: 0 — диалог типа «открыть», 1 — диалог типа «сохранить».
<ИмяФайла>	Имя переменной, содержащей строковое значение с именем файла. В эту же переменную система возвращает имя выбранного файла.
<ИмяНачКаталога>	Имя переменной, содержащей строковое значение с именем начального каталога. В эту же переменную система возвращает имя выбранного каталога.
<ЗаголовокОкна>	Строковое выражение, с помощью которого можно задать заголовок открываемого окна.
<Фильтр>	Строковое выражение, с помощью которого можно задать список фильтров отбора файлов. Фильтр состоит из двух частей, разделенных сиволом "" (вертикальная черта): строки представления, которая отображается в окне выбора файла, и непосредственно строки маски. В строке маски можно использовать символ "*", что означает наличие любого числа произвольных символов. Символ "?" в строке маски означает наличие одного произвольного символа. Одновременно можно задавать несколько фильтров в виде списка, в качестве разделителей используется символ "|" (вертикальная черта). Например, список из двух фильтров: "Текст(*.txt)|*.txt| Таблицы(*.mxl)|*.mxl"
<Расширение>	Строковое выражение, с помощью которого можно задать расширение файла по умолчанию, которое используется системой при записи файла.
<Таймаут>	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя.

Возвращаемое значение:

Число: 0 — если в окне диалога нажата кнопка «Отмена»; 1 — если в окне диалога нажата кнопка «ОК», при этом в переменную <ИмяФайла> возвращается выбранное имя файла, а в переменную <ИмяНачКаталога> возвращается имя выбранного каталога; -1 (минус единица) — закончилось время <Таймаут> ожидания отклика пользователя.

Описание:

Метод ВыбратьФайл открывает окно диалога выбора/сохранения файла.

Пример:

* В данном примере приведена процедура вызова диалога выбора файла.

```
Процедура УдалФ()
   ИмяВыбрФайла = "";
   ИмяПути="";
   Если ФС.ВыбратьФайл(0, ИмяВыбрФайла, ИмяПути, "Выберите файл",
           "Все файлы (*.*)!*.*", , ) = 1 Тогда
```

```
      ФС.УдалитьФайл(ИмяПути + ИмяФайла);
   КонецЕсли;
КонецПроцедуры
```

ВыбратьФайлКартинки

Открывает окно диалога выбора/сохранения файла картинки.

Синтаксис:

```
ВыбратьФайлКартинки(<ТипДиалога>, <ИмяФайла>, <ИмяНачКаталога>, <ЗаголовокОкна>,
                    <Расширение>, <Таймаут>)
```

Англоязычный синоним:

`SelectPictFile`

Параметры:

`<ТипДиалога>`	Числовое выражение, значение которого определяет тип открываемого диалога. Допустимые значения: 0 — диалог типа «открыть», 1 — диалог типа «сохранить».
`<ИмяФайла>`	Имя переменной, содержащей строковое значение с именем файла. В эту же переменную система возвращает имя выбранного файла.
`<ИмяНачКаталога>`	Имя переменной, содержащей строковое значение с именем начального каталога. В эту же переменную система возвращает имя выбранного каталога.
`<ЗаголовокОкна>`	Строковое выражение, с помощью которого можно задать заголовок открываемого окна.
`<Расширение>`	Строковое выражение, с помощью которого можно задать расширение файла по умолчанию, которое используется системой при записи файла.
`<Таймаут>`	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя.

Возвращаемое значение:

Число: 0 — если в окне диалога нажата кнопка «Отмена»; 1 — если в окне диалога нажата кнопка «ОК», при этом в переменную `<ИмяФайла>` возвращается выбранное имя файла, а в переменную `<ИмяНачКаталога>` возвращается имя выбранного каталога; -1 (минус единица) — закончилось время `<Таймаут>` ожидания отклика пользователя.

Описание:

Метод `ВыбратьФайлКартинки` открывает окно диалога (с возможностью предварительного просмотра) выбора/сохранения файла картинки.

Пример:

* В данном примере приведена процедура вызова диалога выбора файла картинки.

```
// Фото — Объект типа "Картинка"
Процедура ЗагрКарт()
   ИмяВыбрФайла = "";
   ИмяПути = "";
   // Выбор файла с просмотром
   Если ФС.ВыбратьФайлКартинки(0, ИмяВыбрФайла, ИмяПути,
         "Выберите файл", "bmp", ) = 1 Тогда
      Фото.Загрузить(ИмяПути + ИмяВыбрФайла);
   КонецЕсли;
КонецПроцедуры
```

ВыбратьКаталог

Открывает окно диалога выбора каталога.

Синтаксис:

```
ВыбратьКаталог(<ИмяКаталога>, <Заголовок>, <Таймаут>)
```

Англоязычный синоним:

`SelectDirectory`

Параметры:

`<ИмяКаталога>`	Имя переменной, содержащей строковое значение с именем начального каталога. В эту же переменную система возвращает имя выбранного каталога.
`<Заголовок>`	Строковое выражение, с помощью которого можно задать заголовок открываемого окна.
`<Таймаут>`	Необязательный параметр. Числовое выражение, значение которого задает время ожидания системы (в секундах) на отклик пользователя.

Возвращаемое значение:

Число: 0 — если в окне диалога нажата кнопка «Отмена»; 1 — если в окне диалога нажата кнопка «ОК», при этом в переменную `<ИмяНачКаталога>` возвращается имя выбранного каталога; -1 (минус единица) — закончилось время `<Таймаут>` ожидания отклика пользователя.

Описание:

Метод `ВыбратьКаталог` открывает окно диалога выбора каталога.

Пример:

* В данном примере приведена процедура вызова диалога выбора каталога.

```
Функция УстКат(ИмяПути)
   Если ФС.ВыбратьКаталог(ИмяПути, "Выберите каталог", 10) = 1 Тогда
      Возврат ИмяПути;
   Иначе
```

```
      Возврат КаталогПользователя();
   КонецЕсли;
КонецПроцедуры
```

СуществуетФайл

Проверяет существование файла.

Синтаксис:

`Существуетфайл(<ИмяФайла>)`

Англоязычный синоним:

`ExistFile`

Параметры:

`<ИмяФайла>`	Строковое выражение с именем файла.

Возвращаемое значение:

Число: 1 — файл существует; 0 — не существует.

Описание:

Метод `СуществуетФайл` проверяет существование файла.

Пример:

```
Процедура СущФ(ИмяФайла)
   Если ФС.СуществуетФайл(ИмяФайла) = 1 Тогда
      Сообщить("Файл " + ИмяФайла + " — существует");
   КонецЕсли;
КонецПроцедуры
```

КопироватьФайл

Копирует файл.

Синтаксис:

`КопироватьФайл(<ИмяФайлаИсточника>, <ИмяФайлаПриемника>, <ФлагОтказаПерезаписи>)`

Англоязычный синоним:

`FileCopy`

Параметры:

`<ИмяФайлаИсточника>`	Строковое выражение с именем файла источника.
`<ИмяФайлаПриемника>`	Строковое выражение с именем файла приемника.
`<ФлагОтказаПерезаписи>`	Числовое выражение: 0 — при существовании файла приемника заменяет его на копируемый; 1 — при существовании файла приемника копирования не производится.

Описание:

Метод `КопироватьФайл` копирует файл-источник в файл-приемник.

Пример:

* В данном примере приведена процедура копирования файла.

```
Процедура КопирФ()
   ФС.КопироватьФайл(ИмяВыбрФайла, ФПриемник, 1);
КонецПроцедуры
```

УдалитьФайл

Удалить файл.

Синтаксис:

`УдалитьФайл(<ИмяФайла>)`

Англоязычный синоним:

`DeleteFile`

Параметры:

`<ИмяФайла>`	Строковое выражение с именем удаляемого файла.

Описание:

Метод `УдалитьФайл` удаляет указанный файл.

Пример:

* В данном примере приведена процедура удаления файла.

```
Процедура УдалФ()
   ФС.УдалитьФайл(ИмяФайла);
КонецПроцедуры
```

ПереименоватьФайл

Переименовать файл.

Синтаксис:

`ПереименоватьФайл(<ИмяФайлаИсточника>, <ИмяФайлаПриемника>, <ФлагПерезаписи>)`

Англоязычный синоним:

`MoveFile`

Параметры:

`<ИмяФайлаИсточника>`	Строковое выражение с именем файла источника.

<ИмяФайлаПриемника>	Строковое выражение с требуемым именем файла.
<ФлагПерезаписи>	Числовое выражение: 0 — запрещает перемещение файла между дисками и при существовании файла приемника копирования не производится; 1 — разрешает перемещение файла (только для файлов) между дисками и при существовании файла приемника замещает его на копируемый.

Описание:

Метод ПереименоватьФайл выполняет переименование (перемещение) указанного файла.

Пример:

* В данном примере приведена процедура переименования файла.

```
Процедура ПереимФ()
   ФС.ПереименоватьФайл(Результат, ФПриемник, 1);
КонецПроцедуры
```

НайтиПервыйФайл

Открывает выборку файлов по заданной маске и находит первый файл.

Синтаксис:

```
НайтиПервыйФайл(<МаскаИмени>)
```

Англоязычный синоним:

```
FindFirstFile
```

Параметры:

<МаскаИмени>	Строковое выражение с маской имен файлов. В строке маски можно использовать символ "*", что означает наличие любого числа произвольных символов. Символ "?" в строке маски означает наличие одного произвольного символа.

Возвращаемое значение:

Строка с именем найденного файла.

Описание:

Метод НайтиПервыйФайл находит первый файл и открывает выборку файлов по заданной маске.

Замечание. Методы НайтиПервыйФайл и НайтиСледующийФайл возвращают имена файлов в той же последовательности, как это делает команда Dir операционной системы MS DOS, т. е. если задана маска "*.*"для некорневого каталога, первым возвращаемым значением будет "." — обозначающая ссылку на текущий каталог. Имя следующего файла будет ".." — обозначающее ссылку на каталог уровнем выше. При получении следующих файлов возвращаемым значением будет имя найденного файла.

Пример:

* В данном примере приведена процедура открытия выборки файлов по заданной маске.

```
Процедура ПервФ()
   Результат = ФС.НайтиПервыйФайл("*.*");
КонецПроцедуры
```

См. также: НайтиСледующийФайл

НайтиСледующийФайл

Находит следующий файл по открытой выборке файлов.

Синтаксис:

```
НайтиСледующийФайл()
```

Англоязычный синоним:

```
FindNextFile
```

Возвращаемое значение:

Строка с именем найденного файла.

Описание:

Метод НайтиСледующийФайл находит следующий файл по открытой выборке файлов.

Пример:

* В данном примере приведена процедура взятия следующего файла выборки файлов.

```
Процедура СледФ()
   Результат = ФС.НайтиСледующийФайл();
КонецПроцедуры
```

См. также: НайтиПервыйФайл

АтрибутыФайла

Возвращает атрибуты файла.

Синтаксис:

```
АтрибутыФайла(<ИмяФайла>, <РазмерФайла> , <АтрибутыФайла>, <ВремяСоздания>,
            <ВремяПоследнДоступа>, <ВремяПоследнЗаписи>, <РасширенноеИмяФайла>)
```

Англоязычный синоним:

```
GetFileAttr
```

Параметры:

<ИмяФайла>	Строковое выражение с именем файла.
<РазмерФайла>	Возвращаемое Числовое значение размера файла в байтах.
<АтрибутыФайла>	Возвращаемое строковое значение длиной 9 символов, в котором закодированы

	атрибуты файла. Символы могут принимать значения "0" или "1": Первый символ: если "1" — файл только для чтения; Второй символ: если "1" — скрытый файл; Третий символ: если "1" — системный файл; Четвертый символ: если "1" — каталог; Пятый символ: если "1" — архивный файл; Шестой символ: если "1" — обычный файл (все другие атрибуты не установлены); Седьмой символ: если "1" — временный файл; Восьмой символ: если " 1" — файл, сжатый каким-либо архиватором; Девятый символ: если "1" — нет доступа к файлу. Для проверки атрибута можно применять выражение: `Цел(ПолученныйАтрибут/КодАтрибута)/2` где `кодАтрибута` — 1/2/4/16/32... это выражение возвращает 1 или 0, в зависимости от того, установлен атрибут или нет.
`<ВремяСоздания>`	Возвращаемое строковое значение, содержащее дату и время создания файла.
`<ВремяПоследнДоступа>`	Возвращаемое строковое значение, содержащее дату и время последнего доступа к файлу.
`<ВремяПоследнЗаписи>`	Возвращаемое строковое значение, содержащее дату и время последней записи файла.
`<РасширенноеИмяФайла>`	Возвращаемое строковое значение, содержащее полное имя файла.

Описание:
Метод `АтрибутыФайла` считывает атрибуты файла и записывает их значения в передаваемые параметры.
Пример:
* В данном примере приведена процедура считывания атрибутов файла.,

```
Лпроцедура АттрФ()
   ФС.АтрибутыФайла(ИмяВыбрФайла, A, B, C, D, E, F);
КонецПроцедуры
```

СоздатьКаталог

Создает новый каталог файлов.
Синтаксис:
`СоздатьКаталог(<ИмяКат>)`
Англоязычный синоним:
`CreateDirectory`
Параметры:

`<ИмяКат>`	Строковое выражение с именем создаваемого каталога.

Описание:
Метод `СоздатьКаталог` создает новый каталог файлов.
Пример:
* В данном примере приведена процедура создания нового каталога файлов.

```
Процедура СоздДир()
   ФС.СоздатьКаталог(ИмяВыбрДир);
КонецПроцедуры
```

УдалитьКаталог

Удаляет каталог файлов.
Синтаксис:
`УдалитьКаталог(<ИмяКат>)`
Англоязычный синоним:
`RemoveDirectory`
Параметры:
`<ИмяКат>` Строковое выражение с именем удаляемого каталога файлов.
Описание:
Метод `УдалитьКаталог` удаляет каталог файлов.
Пример:
* В данном примере приведена процедура удаления каталога файлов.

```
Процедура УдалДир()
   ФС.УдалитьКаталог(ИмяВыбрДир);
КонецПроцедуры
```

УстТекКаталог

Устанавливает текущий каталог файлов.
Синтаксис:
`УстТекКаталог(<ИмяКат>)`
Англоязычный синоним:
`GetCurrentDirectory`

Параметры:
<ИмяКат> Строковое выражение с именем текущего каталога файлов.
Описание:
Метод УстТекКаталог устанавливает текущий каталог файлов.
Пример:
* В данном примере приведена процедура установки текущего каталога файлов
```
Процедура УстТекДир()
   ФС.УстТекКаталог(ИмяВыбрДир);
КонецПроцедуры
```
См. также: ТекКаталог

ТекКаталог

Возвращает текущий каталог файлов.
Синтаксис:
```
ТекКаталог()
```
Англоязычный синоним:
```
GetCurrentDirectory
```
Возвращаемое значение:
Строковое значение имени текущего каталога файлов.
Описание:
Метод ТекКаталог возвращает строковое значение имени текущего каталога файлов.
Пример:
```
ТекКат = ФС.ТекКаталог();
```
См. также: УстТекКаталог

WindowsКаталог

Возвращает имя Windows директории.
Синтаксис:
```
WindowsКаталог()
```
Англоязычный синоним:
```
GetWindowsDirectory
```
Возвращаемое значение:
Строковое значение — имя Windows директории.
Описание:
Метод WindowsКаталог возвращает имя Windows директории.
Пример:
```
WinKaT = ФС.WindowsКаталог();
```

СвободноеМестоНаДиске

Определить размер свободного дискового пространства.
Синтаксис:
```
СвободноеМестоНаДиске(<ИмяДиска>)
```
Англоязычный синоним:
```
GetDiskFreeSpace
```
Параметры:
<ИмяДиска> Строковое выражение с именем диска.
Возвращаемое значение:
Численое значение — размер свободного дискового пространства в байтах.
Описание:
Метод СвободноеМестоНаДиске возвращает численое значение размера
свободного дискового пространства.
Пример:
* В данном примере приведена процедура определения свободного дискового пространства диска С:
```
Процедура РазмД()
   Рез = ФС.СвободноеМестоНаДиске("C:");
КонецПроцедуры
```

Глава 38
Работа с базами данных формата DBF

Основные понятия

Для облегчения понимания материала данного раздела читателем, не имеющим опыта работы с базами данных, ниже приводятся пояснения терминов, использованных при описании средства встроенного языка для работы с базами данных — агрегатного типа данных XBase.

Поля и записи

Если Вы не знакомы с концепцией базы данных, то ее можно себе представить в виде упорядоченного хранилища информации типа картотеки. Хорошим примером базы данных может служить телефонный справочник организации. Он содержит фамилии, номера телефонов и номера комнат всех сотрудников организации. Каждая строка справочника соответствует одной *записи*, а каждая колонка — *полю*. Каждое поле имеет наименование и характеристики информации для хранения которой оно предназначено: тип, длина, точность. Содержимое поля для конкретной записи называется *значением поля*. Отрывок телефонного справочника, приведенный ниже, иллюстрирует вышесказанное:

	Поле "Фамилия"	**Поле "№ комнаты"**	**Поле "Телефон"**
	Иванов	25	513
Запись № 2 —>	Петров	31	274
	Сидоров	25	513
	…	…	…

Таблица, структура таблицы, файл базы данных

Весь справочник целиком в терминах баз данных называется *таблицей*. Состав входящих в таблицу полей определяет *структуру таблицы*, а состав входящих в таблицу записей — ее содержание. Каждая запись в таблице состоит из того же набора полей, что и таблица целиком, поэтому иногда употребляется термин *структура записи*. Это понятие адекватно структуре таблицы, хотя употребление первого термина представляется более корректным, т.к. таблица имеет структуру независимо от того, имеется ли в ней хотя бы одна запись.

Реализация баз данных формата DBF подразумевает, что каждая таблица хранится в отдельном файле. Поэтому в дальнейшем мы будем применять термин «*файл базы данных*» или «*файл БД*», имея в виду таблицу базы данных.

Индексы, выражения индекса и фильтра

Для организации упорядочивания содержимого файла БД и поиска в ней по значению одного или нескольких полей применяется механизм *индексов*. Его применение можно сравнить с сортировкой картотеки по определенному признаку (совокупности признаков). Однако, в отличие от картотеки, файл БД может иметь сразу несколько индексов, и, соответственно, являться упорядоченным **одновременно** по нескольким признакам. Каждый индекс имеет *наименование*, *признак уникальности*, *выражение индекса* и *фильтр*. Наименование индекса используется для идентификации индекса. Выражение индекса и фильтр представляют собой написанные на специальном языке выражения, вычисление значения которых для каждой записи позволяет определить ее место при упорядочивании и необходимость помещения ее в упорядоченный список (индекс может содержать упоминание не обо всех записях таблицы, а только об удовлетворяющих выражению фильтра). Подробнее выражения индекса и фильтра, применительно к объектам XBase, рассмотрены в конце данного раздела. Уникальный индекс (имеющий установленным признак уникальности) позволяет иметь в индексе ссылки на записи только с различным значением индексного выражения.

Индексный файл

Индексы хранятся в *индексном файле*. Индексный файл может содержать информацию более чем об одном индексе.

Назначение агрегатного типа данных XBase

Для работы с базами данных формата DBF в системе может использоваться специальный агрегатный тип данных — XBase. Механизм работы с базами данных формата DBF предназначен для обеспечения возможности манипулирования ими непосредственно из встроенного языка программы 1С:Предприятие. Каждый XBase-объект может быть связан с одним файлом базы данных.

Атрибуты объекта и поля базы данных

Объекты XBase имеют динамически изменяемый набор атрибутов, состав и тип которых зависит от структуры файла базы данных, с которым связан конкретный объект. Объект содержит в своих атрибутах информацию об одной текущей записи. Каждому полю файла базы данных соответствует один атрибут объекта. Наименование атрибута совпадает с именем поля.

Запись изменений в базу данных

Каждый объект представляет собой структуру данных, расположенных в памяти компьютера и изменение содержимого его атрибутов не вызывает немедленного изменения в файлах базы данных. При включенном режиме авто-

сохранения запись содержимого объекта в файлы БД происходит при изменении позиционирования (переход с следующей записи, поиск по ключу и т. д.), при выключенном режиме автосохранения запись изменений происходит только при вызове соответствующего метода объекта.

Работа с индексными файлами

Следует иметь в виду, что одновременно XBase-объект может быть связан не более, чем с одним индексным файлом. Все изменения в базе данных, сделанные в сеансе работы с одним индексным файлом, никак не отражаются на остальных. Поэтому не рекомендуется иметь более одного индексного файла для БД. В противном случае, после каждого открытия БД с индексным файлом, отличным от открытого в предыдущем сеансе работы с базой, следует производить переиндексацию (обновление содержимого индексного файла).

Удаление записей

Удаление записи из базы данных не приводит к физическому уничтожению ее на диске. В этом случае в специальном служебном поле записи, не доступном обычными средствами, ставится пометка об удалении. На записи, помеченные удаленными, позиционирования не происходит, если не включен специальный режим просмотра удаленнных записей. Имеется набор методов для включения/выключения специального режима просмотра, а также определения, является ли спозиционированная запись удаленной, и восстановления удаленной записи.

Метод сжатия базы вызывает физическое уничтожение записей, помеченных как удаленные. Метод очистки базы вызывает физическое уничтожение всех записей. После применения этих методов восстановление удаленных записей становится невозможным.

Создание базы данных, индекса, индексного файла

Помимо работы с существующими базами данных, агрегатный тип XBase имеет набор методов, позволяющих создать новую базу данных произвольной структуры, новые индексы и новый индексный файл. Следует отметить, что, если использование методов, изменяющих структуру БД, возможно только для объектов, не связанных с базой данных (т. е. для вновь создаваемых БД), то создание новых индексов и индексного файла возможно как для создаваемых БД, так и для уже существующих и открытых.

Ограничения

Основное назначение объектов XBase — организация экспорта-импорта информации в/из внешних файлов формата DBF. Использование методов Xbase для доступа непосредственно к данным информационной базы системы 1С:Предприятие не рекомендуется.

Объекты XBase не поддерживают поля типа memo (см. `ДобавитьПоле`).

Поддерживаемые объектом XBase фукции, которые можно использовать в выражениях и фильтрах индексов перечислены в разделе «Выражение и фильтр индекса».

Объекты XBase поддерживают только монопольный доступ к файлам. XBase-объекты поддерживают индексные файлы в формате CDX. Однако, использование внешними программами (например, FoxBase) индексных файлов, созданных с помощью объектов XBase, так же, как и использование объектами индексных файлов, созданных внешними программами, не рекомендуется из-за возможной несовместимости версий.

Контекст работы с XBase

Во всех программных модулях для работы с базами данных формата DBF следует использовать объекты типа XBase. Можно создать произвольное число объектов типа XBase при помощи функции `СоздатьОбъект`. Чтобы обратиться к атрибуту объекта или вызвать его метод, имя атрибута или метода (с указанием необходимых параметров) пишется через точку после имени объекта.

Русскоязычное написание ключевого слова `XBase` отсутствует.

Пример:

```
ДБФ = СоздатьОбъект("XBase");
ИмяФайла = "";
ИмяПути = "";
ФС.ВыбратьФайл(0, ИмяФайла, ИмяПути, , "*.*", , );
ПутьДБ = ИмяПути + ИмяФайла;
ДБФ.ОткрытьФайл(ПутьДБ);
Если ДБФ.Открыта() = 1 Тогда
   Предупреждение("База открыта", 2);
   ДБФ.Первая();
   ФИО = ДБФ.FIO;      // Подразумевается, что поле FIO имеется в открытой БД
   Предупреждение("Первым в " + ПутьДБ + " упомянут т." + ФИО, 5);
Иначе
   Предупреждение("НЕ смогли открыть Базу!", 2);
КонецЕсли;
```

Атрибуты объекта XBase

<Поле>

Предоставляет доступ к полю записи с именем «Поле». Имя поля должно совпадать с именем поля базы данных, с которой связан объект.

Синтаксис:

<Поле>

Описание:

Атрибут <Поле> предоставляет доступ к полю записи с именем «Поле». В тексте программного модуля под термином <Поле> понимается имя конкретного столбца файла базы данных, с которым связан объект.

Атрибут имеет смысл только если файл базы данных находится в открытом состоянии.

Состав атрибутов объекта может динамически изменяться в процессе исполнения 1С:Предприятия и определяется набором полей файла базы данных, с которым объект связан в данных момент времени.

Пример:

```
ФИО = ДБФ.FIO;
```

Ключ

Предоставляет доступ к агрегатному типу данных типа «Ключ».

Синтаксис:

Ключ

Англоязычный синоним:

Key

Описание:

Агрегатный тип данных типа «Ключ» имеет только атрибуты. Состав атрибутов полностью повторяет атрибуты XBase-объекта за исключением того, что отсутствует атрибут «Ключ». Значения атрибутов используются XBase-объектом для вычисления выражения индекса при использовании метода НайтиПоКлючу.

Пример:

```
ДБФ.Ключ.FIO = ФИО;
ЗаписьНайдена = ДБФ.НайтиПоКлючу(0);
```

Методы объекта XBase

СоздатьФайл

Создать новый файл базы данных.

Синтаксис:

СоздатьФайл(<ПутьКБазе>, <ПутьКИндексу>)

Англоязычный синоним:

CreateFile

Параметры:

<ПутьКБазе>	Строковое выражение, содержащее путь к файлу базы данных формата DBF.
<ПутьКИндексу>	Необязательный параметр. Строковое выражение, содержащее путь к индексному файлу базы данных.

Описание:

Метод СоздатьФайл создает новый файл базы данных формата DBF. Если база с таким именем существует, то выдается ошибка.

Пример:

```
ДБФ = СоздатьОбъект("XBase");
// Определение полей и индексов новой БД
.....................
.....................
// Теперь физически создаем базу
ДБФ.СоздатьФайл("mydb.dbf", "mydb.cdx");
```

См. также: ДобавитьПоле, ДобавитьИндекс

ОткрытьФайл

Открыть существующую базу.

Синтаксис:

ОткрытьФайл(<ПутьКБазе>, <ПутьКИндексу>, <ТолькоЧтение>)

Англоязычный синоним:

OpenFile

Параметры:

<ПутьКБазе>	Строковое выражение, содержащее путь к файлу базы данных формата DBF.
<ПутьКИндексу>	Необязательный параметр. Строковое выражение, содержащее путь к индексному файлу базы данных.
<ТолькоЧтение>	Необязательный параметр. Число: 1 — файл открывается в режиме только чтение; 0 — файл открывается в режиме полного доступа (при этом файл открывается в эксклюзив-

ном режиме). Значение по умолчанию — 0.

Описание:

Метод ОткрытьФайл открывает существующую базу данных формата DBF.

Пример:

```
// Подразумевается, что объект ДБФ уже создан и
// доступен в данном контексте
Процедура ОткрДБ()
   ИмяФайла = "";
   ИмяПути = "";
   ФС.ВыбратьФайл(0, ИмяФайла, ИмяПути, , "*.*", , );
   ПутьДБ = ИмяПути + ИмяФайла;
   ДБФ.ОткрытьФайл(ПутьДБ);
   Если ДБФ. Открыта() = 1 Тогда
      Предупреждение("База открыта", 2);
   Иначе
      Предупреждение("Не смогли открыть Базу!", 2);
   КонецЕсли;
КонецПроцедуры
```

Открыта

Прочитать значение флага открытия файла.

Синтаксис:

Открыта()

Англоязычный синоним:

IsOpen

Возвращаемое значение:

Число: 1 — база открыта; 0 — база не открыта.

Описание:

Метод Открыта возвращает значение флага открытия файла базы данных.

Пример:

См. предыдущий пример.

ЗакрытьФайл

Закрыть базу.

Синтаксис:

ЗакрытьФайл()

Англоязычный синоним:

CloseFile

Описание:

Метод ЗакрытьФайл закрывает ранее открытую или созданную базу данных формата DBF.

Пример:

```
Процедура ЗакрДБ()
   Если ДБФ.Открыта() = 1 Тогда
      ДБФ.ЗакрытьФайл();
   КонецЕсли;
КонецПроцедуры
```

ОчиститьФайл

Очистить все записи в базе.

Синтаксис:

ОчиститьФайл()

Англоязычный синоним:

Zap

Описание:

Метод ОчиститьФайл удаляет все записи в базе. При этом все существующие записи удаляются физически и не могут быть впоследствии восстановлены.

Пример:

```
Процедура ОчистДБ()
   Если Вопрос("Вы уверены, что надо очистить базу?", 1, 5) = 1 Тогда
      ДБФ.ОчиститьФайл();
   КонецЕсли;
КонецПроцедуры
```

См. также: Удалить, Восстановить

Сжать

Сжать базу, убрать удаленные записи.

Синтаксис:

Сжать()

Англоязычный синоним:

Pack

Описание:

Метод Сжать уменьшает размер файла базы данных, убирая записи, помеченные как удаленные.

Пример:

```
Процедура СжатьДБ()
   Если Вопрос("Уничтожить записи, помеченные как удаленные?", 1, 5) = 1 Тогда
      ДБФ.Сжать();
   КонецЕсли;
КонецПроцедуры
```

См. также: Удалить, Восстановить

Переиндексировать

Переиндексировать базу.

Синтаксис:

Переиндексировать()

Англоязычный синоним:

Reindex

Описание:

Метод Переиндексировать выполняет переиндексирование базы. Объекты XBase автоматически выполняют индексацию при изменениях базы данных, однако, при работе более чем с одним индексным файлом или после аварийных завершений может потребоваться принудительная переиндексация (для «своих» БД система 1С:Предприятие выполняет это автоматически).

Пример:

```
Процедура ПерИндДБ()
   Если ФлагАварийногоЗавершения = 1 Тогда
      ДБФ.Переиндексировать();
   КонецЕсли;
КонецПроцедуры
```

ПоказыватьУдаленные

Установить режим показа удаленных записей в базе.

Синтаксис:

ПоказыватьУдаленные(<Режим>)

Англоязычный синоним:

ShowDeleted

Параметры:

<Режим>	Необязательный параметр. Числовое выражение: 1 — установить режим показа удаленных записей в базе. 0 — снять режим.

Возвращаемое значение:

Текущее числовое значение режима показа удаленных записей в базе (на момент до исполнения метода).

Описание:

Метод ПоказыватьУдаленные позволяет определить и изменить режим показа удаленных записей в базе. В случае, если параметр не указан, изменения режима не происходит.

Пример:

```
Процедура ПоказатьНомераУдалЗап()
   РежПокУд = ДБФ.ПоказыватьУдаленные(1);
   ДБФ.Первая();
   Пока ДБФ.ВКонце() = 0 Цикл
      Если ДБФ.ЗаписьУдалена() = 1 Тогда
         Предупреждение("Номер удаленной записи – " +
                        Строка(ДБФ.НомерЗаписи()), 3);
      КонецЕсли;
      Если ДБФ.Следующая() = 0 Тогда
         Прервать;
      КонецЕсли;
   КонецЦикла;
   ДБФ.ПоказыватьУдаленные(РежПокУд);
КонецПроцедуры
```

См. также: Удалить, Восстановить

Первая

Перейти на первую запись.

Синтаксис:

Первая()

Англоязычный синоним:

```
First
```

Возвращаемое значение:

Число: 1 — если действие выполнено; 0 — если действие не выполнено.

Описание:

Метод `Первая` предназначен для перехода на первую запись. Если не установлен текущий индекс, объект позиционируется на первую запись в базе данных, если установлен — на запись, имеющую самое младшее значение выражения текущего индекса (если индекс создавался с установленным флагом «Убывание», то наоборот).

Пример:

```
// Обнуляет значения всех полей первой записи
Процедура ОчистПервуюЗап()
   ДБФ.Первая();
   ДБФ.Очистить();
   ДБФ.Записать();
КонецПроцедуры
```

Последняя

Перейти на последнюю запись.

Синтаксис:

```
Последняя()
```

Англоязычный синоним:

```
Last
```

Возвращаемое значение:

Число: 1 — если действие выполнено; 0 — если действие не выполнено.

Описание:

Метод `Последняя` предназначен для перехода на последнюю запись. Если не установлен текущий индекс, объект позиционируется на последнюю запись в базе данных, если установлен — на запись, имеющую самое старшее значение выражения текущего индекса (если индекс создавался с установленным флагом «Убывание», то наоборот).

Пример:

```
// Вычисляет сумму полей AMOUNT всех существующих записей,
// начиная с последней записи и двигаясь к первой
Функция ВычислитьСумму()
   ДБФ.Последняя();
   Сумма = 0;
   Пока ДБФ.Вначале() = 0 Цикл
      Сумма = Сумма + ДБФ.AMOUNT;
      ДБФ.Предыдущая();
   КонецЦикла;
   Возврат Сумма;
КонецФункции
```

Следующая

Перейти на следующую запись.

Синтаксис:

```
Следующая()
```

Англоязычный синоним:

```
Next
```

Возвращаемое значение:

Число: 1 — получена следующая запись; 0 — следующая запись не найдена.

Описание:

Метод `Следующая` предназначен для позиционирования объекта на следующую запись. В зависимости от того, установлен ли текущий индекс, позиционирование происходит либо в порядке возрастания физического номера записи, либо в порядке возрастания значения выражения текущего индекса (если индекс создавался с установленным флагом "Убывание", то в порядке убавания).

Пример:

```
// Обнуляет значения всех полей всех существующих записей
Процедура ОчистЗап() ДБФ.Первая();
   Пока 1 = 1 Цикл
      ДБФ.Очистить();
      ДБФ.Записать();
      Если ДБФ.Следующая() = 0 Тогда
         Возврат;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

Предыдущая

Перейти на предыдущую запись.

Синтаксис:
Предыдущая()
Англоязычный синоним:
Prev
Возвращаемое значение:
Число: 1 — получена предыдущая запись; 0 — предыдущая запись не найдена.
Описание:
Метод Предыдущая предназначен для перехода на предыдущую запись. В зависимости от того, установлен ли текущий индекс, позиционирование происходит либо в порядке убывания физического номера записи, либо в порядке убывания значения выражения текущего индекса (если индекс создавался с Установленным флагом «Убывание», то в порядке возрастания).
Пример:
См. пример метода Последняя

НомерЗаписи

Возвращает значение номера текущей записи.
Синтаксис:
НомерЗаписи()
Англоязычный синоним:
RecNo
Возвращаемое значение:
Числовое значение — номер текущей записи.
Описание:
Метод НомерЗаписи возвращает физический порядковый номер текущей записи в файле базы данных. Результат не зависит от установки текущего индекса.
Пример:
```
Процедура ПоказатьНомераУдалЗап()
   РежПокУд = ДБФ.ПоказыватьУдаленные(1);
   ДБФ.Первая();
   Пока ДБФ.Вконце() = 0 Цикл
      Если ДБФ.ЗаписьУдалена() = 1 Тогда
         Стр1 = "Номер удаленной записи — ";
         Стр2 = Строка(ДБФ.НомерЗаписи());
         Предупреждение(Стр1 + Стр2, 3);
      КонецЕсли;
      Если ДБФ.Следующая() = 0 Тогда
         Прервать;
      КонецЕсли;
   КонецЦикла;
   ДБФ.ПоказыватьУдаленные(РежПокУд);
КонецПроцедуры
```
См. также: Перейти

Перейти

Перейти на запись по ее номеру записи.
Синтаксис:
Перейти(<НомерЗаписи>)
Англоязычный синоним:
GoTo
Параметры:
<НомерЗаписи> Числовое выражение — физический порядковый номер записи в базе данных.
Описание:
Метод Перейти позволяет перейти на запись по ее физическому порядковому номеру в базе данных. Позволяет перейти на запись, помеченную удаленной, даже если не установлен режим просмотра удаленных записей.
Пример:
```
Процедура ИнформацияОЗаписи(НЗап)
   ДБФ.Перейти(НЗап);
   СтрЗап = "н." + ДБФ.НомерЗаписи();
   СтрЗап = СтрЗап + " Уд. " + ДБФ.ЗаписьУдалена();
   СтрЗап = СтрЗап + " Содержит -";
   Для КП = 1 По ДБФ.КоличествоПолей() Цикл
      СтрЗап = СтрЗап + Строка(ДБФ.ПолучитьЗначениеПоля(КП)) + "; ";
   КонецЦикла;
   Предупреждение(СтрЗап, 10);
КонецПроцедуры
```
См. также: НомерЗаписи

ВКонце

Прочитать значение флага конца файла базы банных.

Синтаксис:

```
ВКонце()
```

Англоязычный синоним:

```
EOF
```

Возвращаемое значение:

Число: 1 — указатель в файле находится за последней записью; 0 — конец файла не достигнут.

Описание:

Метод `ВКонце` возвращает значение флага конца файла.

Пример:

```
Функция МаксимальнаяПокупка()
   ДБФ.Первая();
   Уплачено = 0;
   Пока ДБФ.Вконце() = 0 Цикл
      Если ДБФ.AMOUNT > Уплачено Тогда
         Уплачено = ДБФ.AMOUNT;
      КонецЕсли;
      ДБФ.Следующая();
   КонецЦикла;
   Возврат Уплачено;
КонецПроцедуры
```

ВНачале

Прочитать значение флага начала файла.

Синтаксис:

```
ВНачале()
```

Англоязычный синоним:

```
BOF
```

Возвращаемое значение:

Число: 1 — указатель в файле находится перед первой записью; 0 — начало файла не достигнуто.

Описание:

Метод `ВНачале` возвращает значение флага начала файла.

Пример:

```
Функция СреднСтоимПокупки()
   ДБФ.Последняя();
   Уплачено = 0;
   Записей = 0;
   Пока ДБФ.ВКонце() = 0 Цикл
      Уплачено = Уплачено + ДБФ.AMOUNT;
      Записей = Записей + 1;
      ДБФ.Следующая();
   КонецЦикла;
   Если Записей > 0 Тогда
      Уплачено = Уплачено / Записей;
   КонецЕсли;
   Возврат Уплачено;
КонецПроцедуры
```

ТекущийИндекс

Установить/определить текущий индекс.

Синтаксис:

```
ТекущийИндекс(<НазваниеИндекса>)
```

Англоязычный синоним:

```
CurrentIndex
```

Параметры:

`<НазваниеИндекса>`	Необязательный параметр. Строковое выражение с названием индекса. Если параметр опущен, изменения текущего индекса не происходит.

Возвращаемое значение:

Строковое значение с названием текущего индекса (на момент до выполнения метода).

Описание:

Метод `ТекущийИндекс` устанавливает/определяет текущий индекс работы с базой. Установка текущего индекса работы с базой оказывает влияние на работу методов `Найти`, `НайтиПоКлючу`, `Первая`, `Последняя`, `Следующая`, `Предыдущая`.

Пример:

```
Процедура НайтиМладшийПоИндексу(Инд)
   ИмяТекИнд = ДБФ.ТекущийИндекс(Инд);
```

```
    ДБФ.Первая();
    ДБФ.ТекущийИндекс(ИмяТекИнд);
КонецПроцедуры
```

Найти

Найти запись по индексу.

Синтаксис:

`Найти(<Ключ>, <Режим>)`

Англоязычный синоним:

`Find`

Параметры:

`<Ключ>`	Строковое выражение со значением выражения текущего индекса.
`<Режим>`	Числовое выражение. Режим поиска записей: 0 — ищет запись на точное соответствие ключу (=); 1 — ищет запись на точное соответствие с ключом или большую (>=); 2 — ищет запись с большим ключом (>); -1 (минус единица) — ищет запись на точное соответствие с ключом или меньшую(<=); -2 (минус два) — ищет запись с меньшим ключом (<).

Возвращаемое значение:

Число: 1 — если действие выполнено (запись найдена); 0 — если действие не выполнено.

Описание:

Метод `Найти` позволяет найти запись, соответствующую данному значению `<Ключ>` ключа по текущему индексу и режиму поиска, переданному в качестве параметра. Указатель устанавливается на найденную запись.

Следует отметить, что вычисление значения индекса при работе XBase-объекта с базой данных производится объектом самостоятельно на основании выражения, переданного ему при создании ключа (см. описание метода `ДобавитьИндекс`). При вызове данного метода значение `<Ключ>` должно быть вычислено средствами встроенного языка, что может представлять определенные трудности, если индекс составной, поля, включенные в выражение индекса, отличны от строковых, и т. д. Поэтому применимость данного метода ограничена случаем использования простых строковых выражений индекса. В более сложных случаях следует применять универсальный метод `НайтиПоКлючу`.

Пример:

```
ЗаписьНайдена = ДБФ.Найти("Иванов", 0);
```

НайтиПоКлючу

Найти запись по индексу.

Синтаксис:

`НайтиПоКлючу(<Режим>)`

Англоязычный синоним:

`FindByKey`

Параметры:

`<Режим>`	Числовое выражение. Режим поиска записей: 0 — ищет запись на точное соответствие ключу (=); 1 — ищет запись на точное соответствие с ключом или большую (>=); 2 — ищет запись с большим ключом (>); -1 (минус единица) — ищет запись на точное соответствие с ключом или меньшую(<=); -2 (минус два) — ищет запись с меньшим ключом (<).

Возвращаемое значение:

Число: 1 — если действие выполнено (запись найдена); 0 — если действие не выполнено.

Описание:

Метод `НайтиПоКлючу` позволяет найти запись, соответствующую значениям атрибутов агрегатного объекта типа «Ключ» XBase-объекта по текущему индексу и режиму поиска, переданному в качестве параметра. Указатель устанавливается на найденную запись. Перед вызовом метода следует установить значения всех атрибутов агрегатного объекта типа «Ключ», которые участвуют в вычислении выражения текущего индекса (см. описание метода `ДобавитьИндекс`).

Пример:

```
ДБФ.Ключ.Name = "Иванов";
ДБФ.Ключ.DIV_ID = 15; // отдел новых разработок
ЗаписьНайдена = ДБФ.НайтиПоКлючу(0);  // поиск по составному, неоднородному ключу
```

ПолучитьЗначениеПоля

Получить значение поля записи.

Синтаксис:

`ПолучитьЗначениеПоля(<НазваниеПоля>)`

Англоязычный синоним:

`GetFieldValue`

Параметры:

`<НазваниеПоля>`	Строковое выражение с названием поля или числовое выражение с номером поля.

Возвращаемое значение:

Значение поля записи, тип зависит от типа поля.

Описание:

Метод ПолучитьЗначениеПоля позволяет определить значение поля текущей записи. Метод позволяет работать с базами данных неизвестной заранее структуры.

Пример:

```
Процедура ПерНаЗап(НЗап)
   ДБФ.Перейти(НЗап);
   СтрЗап = "" + ДБФ.НомерЗаписи() + "; " + ДБФ.ЗаписьУдалена() + "; ";
   Для КП = 1 По ДБФ.КоличествоПолей() Цикл
      СтрЗап = СтрЗап + Строка(ДБФ.ПолучитьЗначениеПоля(КП)) + "; ";
   КонецЦикла;
КонецПроцедуры
```

УстановитьЗначениеПоля

Установить значение поля.

Синтаксис:

```
УстановитьЗначениеПоля(<НазваниеПоля>, <Значение>)
```

Англоязычный синоним:

```
SetFieldValue
```

Параметры:

<НазваниеПоля>	Строковое выражение с названием поля или числовое выражение с номером поля.
<Значение>	Значение поля.

Описание:

Метод УстановитьЗначениеПоля позволяет установить новое значение атрибута объекта, соответствующего полю текущей записи. Для записи изменений в базу данных необходимо инициировать запись (см. «Назначение агрегатного типа данных Xbase»). Метод позволяет работать с базами данных неизвестной заранее структуры.

Пример:

```
ДБФ.УстановитьЗначениеПоля(НаименованиеПоля, 1234);
```

Добавить

Добавить новую пустую запись.

Синтаксис:

```
Добавить()
```

Англоязычный синоним:

```
Add
```

Описание:

Метод Добавить добавляет новую пустую запись. Атрибуты объекта обнуляются (см. метод Очистить). Для записи изменений в базу данных необходимо инициировать запись (см. «Назначение агрегатного типа данных Xbase»).

Пример:

```
Процедура ДобЗап(Поле!, Поле2)
   ДБФ.Добавить();
   // Определяем поля новой записи
   ДБФ.FIELD1 = Поле1;
   ДБФ.FIELD2 = Поле2;
   ДБФ.Записать();
КонецПроцедуры
```

Скопировать

Скопировать текущую запись.

Синтаксис:

```
Скопировать()
```

Англоязычный синоним:

```
Copy
```

Описание:

Метод Скопировать добавляет новую запись, копирующую текущую запись. Для записи изменений в базу данных необходимо инициировать запись (см. «Назначение агрегатного типа данных Xbase»).

Пример:

```
Процедура КопирЗап()
   ДБФ.Скопировать();
   ДБФ.Записать();
КонецПроцедуры
```

Автосохранение

Установить режим автоматического сохранения изменений в базе.

Синтаксис:

```
Автосохранение(<Режим>)
```

Англоязычный синоним:

AutoSave

Параметры:

<Режим> Необязательный параметр. Числовое выражение: 1 — установить режим автоматического сохранения изменений в базе. 0 — снять режим.

Возвращаемое значение:

Текущее числовое значение режима автоматического сохранения изменений в базе (на момент до исполнения метода).

Описание:

Метод Автосохранение позволяет изменить режим автоматического сохранения изменений в базе.

При установленном режиме автосохранения любые изменения позиционирования объекта приводит к автоматической записи изменений атрибутов (если они происходили с текущей записью) в базу данных. Таким образом, применение метода Записать не имеет смысла при установленном режиме автосохранения. Чтобы отказаться от записи изменений в базу данных следует вызвать метод Отменить — при этом восстанавливаются значения атрибутов объекта до изменений и запись не происходит при изменении позиционирования.

При сброшенном режиме автосохранения записи при изменении позиционирования не происходит — для этого нужно вызвать метод Записать, причем до выполнения позиционирования. В этом режиме не имеет смысла метод Отменить, т. к. достаточно просто не выполнять записи.

Пример:

```
ДБФ.Автосохранение(1);
ДБФ.Новая();
ДБФ.РЮ = "Иванов";
ДБФ.PHONE = 215;
ДБФ.Новая();   // Произошла запись
ДБФ.FIO = "Петров";
ДБФ.PHONE = 215;
ДБФ.Отменить();
ДБФ.Новая(); // Записи не произошло
ДБФ.Автосохранение(0);
ДБФ.FIO = "Петров";
ДБФ.PHONE = 314;
ДБФ.Записать(); // Произошла запись
ДБФ.Новая();
ДБФ.FIO = "Сидоров";
ДБФ.PHONE = 215;
ДБФ.Новая(); // Записи не произошло
```

Записать

Записать изменения в базу.

Синтаксис:

Записать()

Англоязычный синоним:

Save

Описание:

Метод Записать выполняет запись изменений в базу данных. До его вызова все изменения объекта производились только в памяти и будут потеряны при осуществлении перехода к другой записи или закрытии базы данных, если отключен режим автосохранения (см. Автосохранение).

Пример:

См. пример метода Автосохранение

Отменить

Отменить запись изменения в базу.

Синтаксис:

Отменить()

Англоязычный синоним:

Cancel

Описание:

Метод Отменить отменяет запись изменения в базу (см. Автосохранение).

Пример:

см. пример метода Автосохранение

Удалить

Удалить текущую запись.

Синтаксис:

Удалить()

Англоязычный синоним:

Del

Описание:

Метод Удалить ставит пометку «удалена» на текущую запись. Физически запись из базы данных не удаляется и может быть впоследствии восстановлена. Полное удаление записей из БД с освобождением дискового пространства, занятого ими, производится вызовом методов Сжать и ОчиститьФайл.

Пример:

```
Процедура УдалЗап()
   ДБФ.Первая();
   Пока 1 = 1 Цикл
      ДБФ.Удалить();
      ДБФ.Записать();
      Если ДБФ.Следующая() = 0 Тогда
         Возврат;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

ЗаписьУдалена

Возвращает значение флага удаления текущей записи.

Синтаксис:

```
ЗаписьУдалена()
```

Англоязычный синоним:

```
RecDeleted
```

Возвращаемое значение:

Число: 1 — запись удалена; 0 — запись не удалена.

Описание:

Метод ЗаписьУдалена возвращает значение флага удаления текущей записи.

Пример:

```
Процедура ВосстЗап()
   ТекРежим = ДБФ.ПоказыватьУдаленные(1);
   ДБФ.Первая(};
   Пока 1 = 1 Цикл
      Если ДБФ.ЗаписьУдалена() = 1 Тогда
         ДБФ.Восстановить();
         ДБФ.Записать();
      КонецЕсли;
      Если ДБФ.Следующая() = 0 Тогда
         ДБФ.ПоказыватьУдаленные(ТекРежим);
         Возврат;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

Восстановить

Восстановить текущую запись.

Синтаксис:

```
Восстановить()
```

Англоязычный синоним:

```
Recall
```

Описание:

Метод Восстановить восстанавливает текущую запись.

Пример:

См. пример метода Удалить

Очистить

Очистить текущую запись. Обнуляет все атрибуты объекта.

Синтаксис:

```
Очистить()
```

Англоязычный синоним:

```
Clear
```

Описание:

Метод Очистить обнуляет все атрибуты объекта. Атрибуты, соответствующие полям типа строковый приобретают значение «пустая строка», числовой — 0, логический — 0, дата — «пустая дата».

Пример:

```
Процедура ОчистЗап()
   ДБФ.Первая();
   Пока 1=1 Цикл
      ДБФ.Очистить();
      ДБФ.Записать();
```

```
      Если ДБФ.Следующая() = 0 Тогда
         Возврат;
      КонецЕсли;
   КонецЦикла;
КонецПроцедуры
```

КоличествоЗаписей

Возвращает количество записей в базе.

Синтаксис:

`КоличествоЗаписей()`

Англоязычный синоним:

`RecCount`

Возвращаемое значение:

Числовое значение — количество записей в базе.

Описание:

Метод `КоличествоЗаписей` возвращает количество записей в базе вместе с записями, помеченными, как удаленные.

Пример:

```
Предупреждение("Записей — " + Строка(ДБФ.КоличествоЗаписей()));
```

КоличествоПолей

Возвращает количество полей базы.

Синтаксис:

`КоличествоПолей()`

Англоязычный синоним:

`FieldCount`

Возвращаемое значение:

Числовое значение — количество полей базы.

Описание:

Метод `КоличествоПолей` возвращает количество полей базы. Может быть использован при работе с базой данных неизвестной заранее структуры.

Пример:

```
Процедура ОписПол()
   Перем Назв;
   Перем Тип;
   Перем Длин;
   Перем Точн;
   СпЗнач = СоздатьОбъект("СписокЗначений");
   СпЗнач.УдалитьВсе();
   Для КП = 1 По ДБФ.КоличествоПолей() Цикл
      ДБФ.ОписаниеПоля(КП, Назв, Тип, Длин, Точн);
      СпЗнач.ДобавитьЗначение("" + КП + "; " + Назв + "; " + Тип + "; " + Длин +
                              "; "+Точн);
   КонецЦикла;
КонецПроцедуры
```

КоличествоИндексов

Возвращает количество индексов в открытом индексном файле.

Синтаксис:

`КоличествоИндексов()`

Англоязычный синоним:

`IndexCount`

Возвращаемое значение:

Числовое значение — количество индексов в открытом индексном файле.

Описание:

Метод `КоличествоИндексов` возвращает количество индексов в открытом индексном файле. Может быть использован при работе с базой данных неизвестной заранее структуры.

Пример:

```
Процедура ОписИнд()
   Перем Назв;
   Перем Выр;
   Перем Уник;
   Перем Убыв;
   Перем Филт;
   СпЗнач = СоздатьОбъект("СписокЗначений");
   СпЗнач.УдалитьВсе();
   Для КП = 1 По ДБФ.КоличествоИндексов() Цикл
```

```
        ДБФ.ОписаниеИндекса(КП.Назв, Выр, Уник, Убыв, Филт);
        СпЗнач.ДобавитьЗначение("" + КП + "; " + Назв + "; " + Выр + "; " + Уник +
                                "; " + Убыв + "; " + Филт);
    КонецЦикла;
КонецПроцедуры
```

ОписаниеПоля

Возвращает описание поля.

Синтаксис:

`ОписаниеПоля(<НомерПоля>, <НазваниеПоля>, <Тип>, <Длина>, <Точность>)`

Англоязычный синоним:

`GetFieldInfo`

Параметры:

`<НомерПоля>`	Числовое выражение — номер поля, для которого требуется получить описание.
`<НазваниеПоля>`	Идентификатор переменной, в которую данный метод вернет строковое значение названия поля.
`<Тип>`	Идентификатор переменной, в которую данный метод вернет числовое значение типа поля. Допустимые значения см. в описании метода `ДобавитьПоле`.
`<Длина>`	Идентификатор переменной, в которую данный метод вернет числовое значение — общую длину поля.
`<Точность>`	Идентификатор переменной, в которую данный метод вернет числовое значение — длину поля после десятичной точки (только для числовых полей).

Описание:

Метод `ОписаниеПоля` возвращает в параметрах `<НазваниеПоля>`, `<Тип>`, `<Длина>`, `<Точность>` описание поля с номером `<НомерПоля>`.

Пример:

См. пример метода `КоличествоПолей`

ОписаниеИндекса

Возвращает описание индекса.

Синтаксис:

```
ОписаниеИндекса(<НомерИндекса>, <НазваниеИндекса>, <Выражение>, <Уникальность>,
                <Убывание>, <Фильтр>)
```

Англоязычный синоним:

`GetIndexInfo`

Параметры:

`<НомерИндекса>`	Числовое выражение — номер индекса, для которого требуется получить описание.
`<НазваниеИндекса>`	Идентификатор переменной, в которую данный метод вернет строковое значение названия индекса.
`<Выражение>`	Идентификатор переменной, в которую данный метод вернет строковое значение — выражение индекса.
`<Уникальность>`	Идентификатор переменной, в которую данный метод вернет числовое значение — флаг уникальности индекса. 1 — уникальный. 0 — не уникальный.
`<Убывание>`	Идентификатор переменной, в которую данный метод вернет числовое значение — флаг направления убывания индекса. 1 — индекс по убыванию значения ключа. 0 — по возрастанию значения ключа.
`<Фильтр>`	Идентификатор переменной, в которую данный метод вернет строковое значение — фильтр индекса.

Описание:

Метод `ОписаниеИндекса` возвращает в параметрах `<НазваниеИндекса>`, `<Выражение>`, `<Уникальность>`, `<Убывание>`, `<Фильтр>` описание индекса с номером `<НомерИндекса>`. Выражение индекса и фильтр индекса описаны в разделе «Выражение и фильтр индекса».

Пример:

См. пример метода `КоличествоИндексов`

НомерПоля

Определить номер поля по названию.

Синтаксис:

`НомерПоля(<НазваниеПоля>)`

Англоязычный синоним:

`FieldNo`

Параметры:

`<НазваниеПоля>`	Строковое выражение с названием поля.

Возвращаемое значение:

Числовое значение номера поля.

Описание:

Метод НомерПоля предназначен для определения номера поля по его на-^анию.

Пример:

```
НомП = ДБФ.НомерПоля("CODE");
```

ДобавитьПоле

Добавить поле в структуру базы.

Синтаксис:

```
ДобавитьПоле(<Название>, <Тип>, <Длина>, <Точность>)
```

Англоязычный синоним:

```
AddField
```

Параметры:

<Название>	Строковое выражение — имя создаваемого поля.
<Тип>	Строковое или числовое выражение — тип создаваемого поля. Допустимые значения: 1 или"N" — число; 2 или "S" — строка; 3 или "D" — дата; 4 или "L" — логическое; 5 или "F" — то же, что и 1, т. е. число; 6 или "M" — memo (не поддерживается).
<Длина>	Числовое выражение — общая длина создаваемого поля.
<Точность>	Числовое выражение — длина создаваемого поля после десятичной точки (только для числовых полей).

Описание:

Метод ДобавитьПоле добавляет поле в описание структуры базы. Данный метод можно использовать только перед созданием новой базы.

Пример:

```
ДБФ = СоздатьОбъект("XBase");
ДБФ.ДобавитьПоле("CODE", 1, 19, 3);
ДБФ.ДобавитьПоле("NAME", 2, 25, 0);
ДБФ.СоздатьФайл("mydb.dbf");
```

См. также: СоздатьФайл, ДобавитьИндекс

ДобавитьИндекс

Добавить индекс в структуру базы.

Синтаксис:

```
ДобавитьИндекс(<Название>, <Выражение>, <Уникальность>, <Убывание>, <Фильтр>)
```

Англоязычный синоним:

```
AddIndex
```

Параметры:

<Название>	Строковое выражение — имя создаваемого индекса.
<Выражение>	Строковое выражение — выражение индекса.
<Уникальность>	Числовое выражение — флаг уникальности индекса. 1 — уникальный. 0 — не уникальный.
<Убывание>	Числовое выражение — флаг направления убывания индекса: 1 — индекс по убыванию значения ключа. 0 — по возрастанию значения ключа.
<Фильтр>	Строковое выражение — фильтр индекса.

Описание:

Метод ДобавитьИндекс добавляет индекс в описание структуры базы. Выражение индекса и фильтр индекса описаны в разделе «Выражение и фильтр индекса».

Пример:

```
ДБФ = СоздатьОбъект("XBase");
ДБФ.ДобавитьПоле("CODE", 1, 19, 3);
ДБФ.ДобавитьПоле("NAME", 2, 25, 0);
ДБФ.ДобавитьИндекс("IDXCODE", "CODE", 1, 0, "");
ДБФ.ДобавитьИндекс("IDXNAME", "NAME", 0, 0, "");
ДБФ.СоздатьФайл("mydb.dbf", "mydb.cdx");
```

См. также: СоздатьФайл, ДобавитьПоле

СоздатьИндексныйФайл

Создать индексный файл.

Синтаксис:

```
СоздатьИндексныйФайл(<ИмяФайла>)
```

Англоязычный синоним:

```
CreateIndex
```

Параметры:

<ИмяФайла>	Строковое выражение — имя индексного файла.

Описание:

Метод `СоздатьИндексныйФайл` создает индексный файл. Создаваемый индексный файл будет содержать все индексы, которые были созданы на текущий момент методом `ДобавитьИндекс`.

Пример:

```
ДБФ.ДобавитьИндексСЧОХСОПЕ" , "CODE" , 1, 0, "" );
ДБФ.ДобавитьИндекс("IDXROOM", "ROOM+FIO", 0, 0, "");
ДБФ.СоздатьИндексныйФайл("IDXCNEW.CDX");
```

См. также: `СоздатьФайл`, `ДобавитьИндекс`, `ОткрытьФайл`

КодоваяСтраница

Установить режим кодировки.

Синтаксис:

```
КодоваяСтраница(<Режим>)
```

Англоязычный синоним:

```
SetCodePage
```

Параметры:

`<Режим>`	Необязательный параметр. Числовое выражение: 0 — Windows-кодировка; 1 — DOS-кодировка. Если параметр не задан, то режим кодировки не меняется (используется для определения текущего режима кодировки без его смены).

Возвращаемое значение:

Текущее числовое значение режима кодировки (на момент до исполнения метода).

Описание:

Метод `КодоваяСтраница` позволяет установить режим кодировки для чтения и записи значений строковых полей в файл базы данных.

Замечание. После открытия, XBase-объект всегда находится в режиме DOS-кодировки.

Пример:

```
ДБФ.КодоваяСтраница(1);
```

КодОшибки

Возвращает код последней ошибки.

Синтаксис:

```
КодОшибки()
```

Англоязычный синоним:

```
ErrorCode
```

Возвращаемое значение:

Числовое значение — код последней ошибки.

Описание:

Метод `КодОшибки` возвращает код завершения последней выполненной операции.

Пример:

```
КО = ДБФ.КодОшибки();
```

Возвращаемые коды ошибок:

Код ошибки	Причина ошибки
-10	Ошибка закрытия файла
-20	Ошибка создания файла
-30	Ошибка определения длины файла
-40	Ошибка установки длины файла
-50	Ошибка при попытке заблокировать файл
-60	Ошибка при открытии файла
-70	Ошибка чтения файла
-80	Ошибка удаления файла
-90	Ошибка переименования файла
-100	Ошибка позиционирования в файле
-110	Ошибка снятия блокировки с файла
-120	Ошибка записи в файл
-200	Файл не является базой данных DBF-формата
-210	Неопознанное имя поля
-220	Неопознанный тип поля
-230	Запись слишком длинная
-300	Индексный файл не содержит информации о записи
-310	Нарушение структуры индексного файла
-330	Указанное имя индекса недоступно
-340	Ошибка уникальности индекса
-400	Ожидается запятая или скобка
-410	Выражение не завершено

-422	IIF() требует параметров одинаковой длины
-425	У STR() и SUBSTR.O 2-й и 3-й параметры — константы
-430	Неверное число параметров
-440	Слишком сложное выражение
-450	Пропущена правая скобка
-460	Неверный тип подвыражения
-470	Неопознанная функция
-480	Неопознанный оператор
-490	Неопознанное значение
-500	Выражение не завершено символом двойной кавычки
-920	Недостаточно памяти

Выражение и фильтр индекса

При работе с индексами во время исполнения 1С:Предприятия XBase-объекты производят вычисление выражения индекса и фильтра. Выражение индекса и фильтра — это строковые выражения, составленные на специальном языке объектов XBase.

Выражение индекса используется для вычисления значения ключа для каждой записи базы данных. Результатом вычисления выражение должно быть значение одного из следующих типов: числовое, строковое, дата или булево. Выражение фильтра должно возвращать результат типа булево. Если значение выражения фильтра для конкретной записи базы данных равно истине, информация об этой записи будет включена в индексный файл, в противном случае индексный файл не будет содержать информацию об этой записи и позиционирование на эту запись XBase-объекта с данным текущим индексом производиться не будет.

Атрибуты (кроме "Ключ"), константы и функции могут быть использованы как части выражений. Части выражения могут объединяться с помощью других функций или операторов. Простейшим выражением может быть имя атрибута: "FULL_NAME". В таком случае тип выражения будет соответствовать типу атрибута.

Константы могут иметь числовой, строковый и булевый тип. Строковые константы заключаются в одиночные кавычки ('Строковая константа'). При необходимости включить в состав строковой константы символа двойной кавычки, его следует предварять обратной косой чертой, например, 'Фирма \"Вега\" '. Булевы константы записываются как .TRUE. или .T. для обозначения истины и .FALSE. или .F. в противном случае.

Операторы используются для объединения частей выражения. Части выражения должны иметь тип, соответствующий оператору, например, оператор "/" (деление) работает с двумя числовыми значениями.

Порядок выполнения операторов в выражении соответствует общепринятому — с учетом скобок и приоритетов операторов. Например выражение "1+2*3" возвратит 7, а "(1+2)*3" возвратит 9.

Числовые операторы:

Имя оператора	Обозначение	Приоритет
Сложение	+	5
Вычитание	-	5
Умножение	*	6
Деление	/	6
В степень	** или ^	7

Строковые операторы:

Имя оператора	Обозначение	Приоритет
Объединение 1	+	5
Объединение 2	-	5

Оператор "Объединение 2" немного отличается тем, что пробелы в конце первой строки будут перемещены в конец результата. Например, результатом вычисления выражения " 'Иванов ' + 'И. И.' " будет "Иванов И. И.', а " 'Иванов ' - 'И. И.' " — 'ИвановИ.И. '.

Операторы отношений:

Имя оператора	Обозначение	Приоритет
Равно	=	4
Не равно	<> или #	4
Меньше	<	4
Больше	>	4
Меньше или равно	<=	4
Больше или равно	>=	4
Содержит	$	4

Пример:

" 'CD' $ 'ABCD' " вернет ".T."

" 8 < 7 " вернет ".F."

Логические операторы:

Имя оператора	Обозначение	Приоритет
Отрицание	.NOT.	3
И	.AND.	2
ИЛИ	.OR.	1

Функции, применяемые в выражениях

Функция	Параметр(ы)	Возвращаемое значение
DATE()		Тип даты. Системная дата.
DAY(Date)	Тип даты.	Число. Номер дня в месяце.
DTOC(Date)	Тип даты.	Строка. Представление даты в формате ММ/ДД/ГГ
DTOS(Date)	Тип даты.	Строка. Представление даты в формате ГГГГММДД
IIF(Log, IfTrue, IfFalse)	Log — выражение булева типа. IfTrue, IfFalse — тип определяется во время исполнения. Должны иметь одинаковую длину и тип.	Тип совпадает с типом IfTrue и IfFalse. Возвращает IfTrue, если значение вычисленного выражения Log равно истине, и IfFalse в противном случае.
LTRIM(String)	String — строка.	Строка. Исходная строка без ведущих пробелов.
Month(Date)	Date — дата.	Число. Номер месяца в году.
STOD(String)	String — строка. Представление даты в формате ГГГГММДД	Тип даты. Преобразованное в дату значение исходной строки.
STR(Number, Length, Decim)	Number — число. Преобразуемое число. Length — число. Количество символов в возвращаемой строке, включая дес. точку. Decim — число. Желаемое кол-во знаков после точки.	Строка. Результат преобразования числа в строковое представление.
SUBSTR(String, StartPos, Num)	String — строка. Исходная строка. StartPos — число. Начальная позиция. Num — число. Кол-во символов.	Строка. Подстрока, состоящая из Num символов от начальной позиции исходной строки.
TIME()		Строка. Системное время в представлении ЧЧ:ММ:СС
TRIM(String)	String — строка.	Строка. Исходная строка без заключительных пробелов.
UPPER(String)	String — строка.	Строка. Преобразование к верхнему регистру
VAL(String)	String — строка.	Число. Преобразование строки в число
YEAR(Date)	Date — дата.	Число. Год.

Глава 39
Работа с Метаданными

Глобальный атрибут «Метаданные» является вспомогательным объектом системы, предназначенным для доступа к свойствам структуры метаданных конфигурации. Возможность доступа к метаданным средствами встроенного языка является дополнительной возможностью и применяется в специальных случаях, например, для создания универсальных отчетов или обработок, которые обращаются к объектам различных видов (документам, справочникам и т. п.). В основных алгоритмах конфигурации данная возможность, как правило, не используется.

Контекст работы с метаданными

Работа с метаданными выполняется через глобальный атрибут «Метаданные», имеющий тип «Метаданные». Объект типа «Метаданные» нельзя создать, используя метод `СоздатьОбъект`.

Объект типа «Метаданные» имеет атрибуты для доступа к свойствам объекта метаданных и методы для доступа к массивам подчиненных объектов метаданных. Глобальный атрибут «Метаданные» предоставляет доступ к общим свойствам конфигурации и к основным объектам метаданных (документам, справочникам и т. п.). Полученный из глобального атрибута подчиненный объект метаданных — документ предоставляет в свою очередь доступ к свойствам документа и к объектам метаданных являющихся реквизитами документа.

Значения, имеющие тип «Метаданные», можно напрямую использовать в качестве параметров в тех методах, где требуется строковый идентификатор метаданного, например, в методе справочника: `Вид(НаименовСправочника)`.

Кроме того, объект типа «Метаданные» можно использовать в качестве параметров, в тех методах, где требуется указать тип данных строкой, например в третий параметр метода `ВвестиЗначение(, , , )`, но передавать можно только значения метаданных, описывающие типизированные объекты, например, `Метаданные.Справочник(1).Реквизит(1)`, т. к. реквизит справочника — типизированный объект.

Англоязычный синоним ключевого слова `Метаданные` — `Metadata`.

Пример:

```
ВыбМетодУдаления = Метаданные.НепосрededственноеУдалениеОбъектов;
```

Атрибуты и методы объекта «Метаданные»

В документации не приводится полный перечень возможных значений атрибутов. Полный перечень атрибутов метаданных можно получить, сформировав в конфигураторе текстовый файл «Описание метаданных». Установленные значения атрибутов можно также узнать из текстового описания структуры метаданных.

Атрибуты, являющиеся по сути признаками с двумя возможными значениями имеют числовой тип и принимают значения 0 или 1. Атрибуты, которые могут иметь несколько возможных значений, выдают строку, отражающую установленный вариант. Атрибуты, отражающие свойство метаданных, выбираемое как ссылка на другой объект метаданных (например, журнал документа) имеют тип «Метаданные».

Пример:

```
ВыбМетодУдаления=Метаданные.НепосредственноеУдалениеОбъектов;
```

У объекта «Метаданные» могут существовать методы для доступа к массивам подчиненных метаданных. Например, для глобального атрибута «Метаданные» для обращения к документам используется метод «Документ».

В качестве параметра методов для доступа к массивам подчиненных метаданных передается:

- число — выдает объект метаданных по указанному номеру;
- строка — выдает объект метаданных по указанному идентификатору;
- параметр не указан — выдает количество подчиненных объектов этого типа.

Пример получения списка документов конфигурации:

```
Для Инд = 1 По Метаданные.Документ() Цикл
    Сообщить(Метаданные.Документ(Инд).Идентификатор);
КснецЦикла;
```

У объекта типа «Метаданные» могут существовать атрибуты, содержащие массив ссылок на объекты метаданных, к ним применяются методы `Количество()` и `Получить(Ном)` для перебора ссылок. Например, для граф отбора таким атрибутом является атрибут «Ссылки», позволяющий получить объекты метаданных включенные в данную графу отбора (реквизиты документов и др.).

Пример:

```
Для Инд = 1 До Метаданные.ГрафаОтбора(Идент).Ссылки.Количество() Цикл
    Сообщить(Метаданные.ГрафаОтбора(Идент).
                Ссылки.Получить(Инд).ПолныйИдентификатор());
КонецЦикла;
```

Методы работы с метаданными

Перечень методов метаданных предназначенных для доступа к подчиненным объектам метаданных можно получить, сформировав в конфигураторе текстовый файл «Описание структуры метаданных».

Дополнительные методы работы с метаданными приведены ниже.

Выбран

Проверяет спозиционирован ли объект типа «Метаданные» на конкретном объекте метаданных или нет.

Синтаксис:

```
Выбран()
```

Англоязычный синоним:

```
Selected
```

Возвращаемое значение:

Число: 1 — если объект соответствует объекту метаданных (спозиционирован); 0 — если не соответствует.

Описание:

Метод `Выбран` возвращает число со значением 1 — объект соответствует объекту метаданных (спозиционирован), 0 — если не соответствует. Например, при обращении к массиву подчиненных метаданных по идентификатору, если метаданного с таким идентификатором не существует, возвращается не спозиционированный объект типа «Метаданные».

Пример:

```
Если Метаданные.Справочник("Организации").Выбран() = 1 Тогда
   Сообщить("Есть справочник организаций");
КонецЕсли;
```

Родитель

Возвращает объект метаданных, которому подчинен данный объект.

Синтаксис:

```
Родитель()
```

Англоязычный синоним:

```
Parent
```

Возвращаемое значение:

Объект метаданных, которому подчинен данный объект.

Описание:

Метод `Родитель` возвращает объект метаданных, которому подчинен данный объект.

Пример:

```
Сообщить("Реквизит принадлежит документу" + РеквМД.Родитель().Идентификатор);
```

ПолныйИдентификатор

Возвращает полный идентификатор объекта.

Синтаксис:

```
ПолныйИдентификатор()
```

Англоязычный синоним:

```
FullIdentifier
```

Возвращаемое значение:

Идентификатор объекта с идентификаторами его родителей через точку.

Описание:

Метод `ПолныйИдентификатор` возвращает идентификатор объекта с идентификаторами его родителей через точку, например: `"Документ.Счет.Цена"`.

Пример:

```
Сообщить("Реквизит " + РеквМД.ПолныйИдентификатор());
```

Представление

Возвращает представление объекта.

Синтаксис:

```
Представление()
```

Англоязычный синоним:

```
Present
```

Возвращаемое значение:

Строковое значение представления объекта.

Описание:

Метод `Представление` возвращает синоним объекта, а если он не задан, то идентификатор.

Пример:

Получение списка видов документов:

```
Спис = СоздатьОбъект("СписокЗначений");
Для Инд = 1 По Метаданные.Документ() Цикл
   Идент = Метаданные.Документ(Инд).Идентификатор;
   Предст = Метаданные.Документ(Инд).Представление();
   Спис.ДобавитьЗначение(Идент, Предст);
КонецЦикла;
```

ДлинаПредставленияЗначения

Возвращает длину представления значения.

Синтаксис:

```
ДлинаПредставленияЗначения(<Мин>, <Макс>, <ДлПоУмолч>)
```

Англоязычный синоним:

```
ValuePresentLen
```

Параметры:

<Мин>	Необязательный параметр. Метод не будет возвращать значение меньше указанной в данном параметре величины.
<Макс>	Необязательный параметр. Метод не будет возвращать значение больше указанной в данном параметре величины.
<ДлПоУмолч>	Необязательный параметр. Метод будет возвращать данную величину для метаданных, для которых длина представления не определена. Параметр задает длину для типов данных, таких как Справочник неопределенного типа и аналогичных, у которых не определена длина.

Возвращаемое значение: возвращает длину представления значения.

Описание:

Метод ДлинаПредставленияЗначения для типизированных объектов типа Реквизит и аналогичных, выдает длину представления значения. Данный метод может использоваться, например, для определения ширины столбца в универсальных отчетах, выводящих данные объектов различных видов.

Пример:

```
Ширина =
   Метаданные.Документ(1).РеквизитШапки(2).ДлинаПредставленияЗначения(5, 50, 30)
```

Глава 40
Связь с внешними приложениями посредством механизмов DDE и OLE Automation

При формировании пользовательского интерфейса прикладной задачи довольно часто существуют ситуации, когда для удобства работы пользователя необходимо дать возможность обрабатывать имеющиеся данные с помощью других, имеющихся на компьютере программ.

Одним из примеров использования такой возможности может служить запуск программы Microsoft Excel (непосредственно из программы 1С:Предприятие) с одновременным выводом в нее результатов расчета. Далее пользователь может обрабатывать полученные результаты (просматривать, редактировать, печатать и т. п.) непосредственно в среде Microsoft Excel.

Одну из возможностей запуска и управления внешними приложениями дает механизм OLE Automation. Встроенный язык системы 1С:Предприятие поддерживает технологию управления объектами OLE. Непосредственно при помощи операторов языка можно управлять, например, программами, входящими в Microsoft Office.

Контекст работы с внешними приложениями

Внешнее приложение, с которым предполагается устанавливать связь, должно отвечать следующим условиям: внешнее приложение должно поддерживать механизм OLE Automation и должно быть установлено на том компьютере, где его намечено использовать.

Во всех программных модулях вызов процедур или функций внешнего приложения может выполняться только при помощи переменной со ссылкой на объект, который создается функцией `СоздатьОбъект`. Чтобы вызвать метод внешнего приложения, имя метода (с указанием необходимых параметров) пишется через точку после идентификатора переменной.

При создании переменной со ссылкой на внешнее приложение, в качестве имени объекта в операторе `СоздатьОбъект` должен выступать зарегистрированный OLE идентификатор внешнего приложения.

Зарегистрированный OLE идентификатор внешнего приложения — это уникальный идентификатор программного продукта, который записывается в регистрационную базу Windows при инсталляции программы.

Примеры OLE идентификаторов:

Excel	"Excel.Application"
LotusApproach	"Approach.Application"
MSGraph	"MSGraph.Application"
PowerPoint	"PowerPoint.Application"

Пример:

```
// Создаем объект Excel (запускаем программу Excel)
Excel = СоздатьОбъект("Excel.Application");
// делаем окно программы Excel видимым и активным
Excel.Visible = 1;
```

Методы внешних приложений

Методы внешних приложений, а также синтаксис их использования у каждого приложения свои. Описание процедур и функций внешнего приложений можно получить только в документации данного приложения. Например, практически все программы, входящие в пакет Microsoft Office используют для своего управления версию языка Visual Basic.

Чтобы вызвать метод внешнего приложения в языке системы 1С:Предприятие, имя метода (с указанием необходимых параметров) пишется через точку после имени ссылки на объект внешнего приложения.

Пример:

```
// Создаем объект Excel и присвоим его переменной языка
ОкноExcel = СоздатьОбъект("Excel.Application");
// устанавливаем имя окна Excel
ОкноExcel.Caption = "Отчет";
// создадим новую рабочую книгу
НовыеРабочиеКниги = ОкноExcel.Workbooks;
РабочаяКнига = НовыеРабочиеКниги.Add();
```

Ограничения:

Не поддерживаются значения, передаваемые по ссылке;

Не поддерживаются массивы, в точности `SAFEARRAY`;

Не поддерживаются значения типа `IUnknown`.

Эти ограничения не столь серьезны, т. к. например, все типы данных, с которыми работает Excel и большинство других наиболее распространенных программ, полностью поддерживаются.

Типы поддерживаемых данных:

```
Boolean
Currency
Date
Double-precision floating-point
Integer
```

```
Long integer
Object
Single-precision floating-point
String
```

Пример:

```
//В данном примере запускается программа Excel,
// затем в нее передаются некоторые данные, и затем
// на их основе строится диаграмма.
// Создаем объект Excel и присвоим его переменной языка
ОкноExcel = СоздатьОбъект("Excel.Application");
// устанавливаем имя окна Excel
ОкноExcel.Caption = "Отчет";
// создадим новую рабочую книгу
НовыеРабочиеКниги = ОкноExcel.Workbooks;
РабочаяКнига = НовыеРабочиеКниги.Add();
ЧислоРядов = 10;
ЧислоСтолбцов = 5;
// проставим названия строк
Для Ряд = 1 По ЧислоРядов Цикл
   Ячейка = ОкноExcel.Cells(Ряд + 1, 1);
   Ячейка.Value = "Строка " + Строка(Ряд);
КонецЦикла;
// проставим названия столбцов
Для Столбец = 1 По ЧислоСтолбцов Цикл
   Ячейка = ОкноExcel.Cells(1, Столбец + 1);
   Ячейка.Value = "Столбец " + Строка(Столбец);
КонецЦикла;
// заполним ячейки таблицы значениями
Для Ряд = 1 По ЧислоРядов Цикл
   Для Столбец = 1 По ЧислоСтолбцов Цикл
      Ячейка = ОкноExcel.Cells(Ряд + 1, Столбец + 1);
      Ячейка.Value = Ряд + Столбец;
   КонецЦикла;
КонецЦикла;
// выделим область в таблице и присвоим ее переменной языка
Область = ОкноExcel.Range(ОкноExcel.Cells(1, 1),
                          ОкноExcel.Cells(ЧислоРядов + 1, ЧислоСтолбцов + 1));
// зададим имя выделенной области
Область.Name = "ОбластьДанных";
// определим рамку выделенной области и присвоим ее переменной языка
Рамка = Область.Borders;
// установим стили для рамки выделенной области
Рамка.LineStyle = 1;
Рамка.Colorlndex = 3;
// построим диаграмму Лист=РабочаяКнига.Worksheets(1);
Диаграмма = Лист.ChartObjects();
Диаграмма = Диаграмма.Add(5, 5 + Область.Top + Область.Height,
                          Область.Width, Область.Height);
МояДиаграмма = Диаграмма.Chart;
МояДиаграмма.ChartWizard("ОбластьДанных ", -4102, 6, 1, 1, 1, 1, "Отчет");
// сделаем окно Excel видимым и активным
ОкноExcel.Visible = 1;
```

Работа системы 1С:Предприятие в качестве OLE Automation сервера

Система 1С:Предприятие может быть запущена внешним приложением в качестве OLE Automation сервера и предоставляет доступ ко всем атрибутам и методам своего *глобального контекста* (см. «Контекст выполнения программного модуля»). Кроме того, OLE-сервер 1С:Предприятие имеет дополнительные методы, с помощью которых можно выполнить последовательность операторов или вычислить выражение, заданное на встроенном языке 1С:Предприятие.

Для запуска системы 1С:Предприятие в качестве OLE Automation сервера из внешнего приложения, выполняется следующая последовательность действий:

- создается объект с OLE идентификатором:
  - `V1CEnterprise.Application` — версия независимый ключ;
  - `V77.Application` — версия зависимый ключ;
  - `V77S.Application` — версия зависимый ключ, SQL версия;
  - `V77L.Application` — версия зависимый ключ, локальная версия;
  - `V77M.Application` — версия зависимый ключ, сетевая версия.

- выполняется инициализация системы 1С:Предприятие методом `Initialize`.
- вызываются атрибуты и методы системы 1С:Предприятие как OLE Automation сервера

Замечание 1: Поскольку программа 1С:Предприятие является одновременно OLE Automation сервером и OLE Automation клиентом, то возможно из 1С:Предприятие обращаться к другой копии 1С:Предприятие (например, к другой конфигурации) для обмена данными. В основном все современные программные продукты поддерживают механизм OLE Automation, это касается в частности MS Office, MS FoxPro и приложений на них написанных, DAO и т. п., поэтому программа 1С:Предприятие может полностью интегрироваться с ними.

Замечание 2: Не локализованные версии внешних программ, обращающихся к программе 1С:Предприятие посредством OLE Automation, могут неправильно интерпретировать русские идентификаторы объектов агрегатных типов данных, например, реквизитов справочников. Данное замечание не относится к продуктам MS Office и к программам, использующим в качестве языка обращения к OLE объектам Microsoft Visual Basic. Рекомендуется использовать локализованные версии программных продуктов, либо в конфигурации использовать идентификаторы без символов кириллицы. Для обращения к атрибутам и методам агрегатных типов данных системы 1С:Предприятие из внешних приложений рекомендуется использовать их англоязычные синонимы.

Замечание 3: Все созданные объекты OLE Automation существуют до тех пор, пока существует переменная, которая содержит значение данного объекта. Следовательно, сама программа 1С:Предприятие, выступающая в качестве объекта OLE Automation в другой программе, будет находиться в памяти компьютера до удаления или изменения значения переменной, содержащей ее в качестве объекта.

Атрибуты системы 1С:Предприятие как OLE Automation сервера

Система 1С:Предприятие в качестве OLE Automation сервера предоставляет полный доступ к своему *глобальному контексту* (см. «Контекст выполнения программного модуля»). Поэтому объект OLE-сервер 1С:Предприятие в качестве своих атрибутов может иметь: системные константы, значения заданных в конфигураторе *констант*, *перечислений*, *регистров*, *видов расчета*, *групп видов расчета*, а также переменные, объявленные в глобальном программном модуле с ключевым словом `Экспорт`.

Методы системы 1С:Предприятие как OLE Automation сервера

Система 1С:Предприятие в качестве OLE Automation сервера предоставляет полный доступ к своему *глобальному контексту* (см. «Контекст выполнения программного модуля»). Поэтому объект OLE-сервер 1С:Предприятие в качестве своих методов может иметь: системные процедуры и функции, а также процедуры и функции глобального программного модуля, объявленные с ключевым словом `Экспорт`. Кроме того, OLE-сервер 1С:Предприятие имеет три дополнительных метода: `Initialize`, `EvalExpr`, `ExecuteBatch`.

Initialize

Выполнить инициализацию системы 1С:Предприятие.

Синтаксис:

```
Initialize(<Имя_Объекта>.RMTrade, <КомандныйСтрока>, <ПустаяСтрока>)
```

Англоязычный синоним:

```
Initialize
```

Параметры:

`<Имя_Объекта>`	Идентификатор созданного OLE объекта 1С:Предприятие.
`RMTrade`	Добавочное ключевое слово.
`<КомандныйСтрока>`	Строковое выражение — командная строка запуска 1С:Предприятие.
`<ПустаяСтрока>`	Строковое выражение. Параметр может содержать пустую строку или строковое значение "NO_SPLASH_SHOW" — отключить заставку при запуске системы 1С:Предприятие в режиме OLE Automation сервера.

Возвращаемое значение:

Значение логического типа: `TRUE`, если инициализация прошла удачно, или `FALSE`, если нет.

Замечание: В OLE Automation TRUE и FALSE имеют соответственно значения: -1 (минус единица) и 0.

Описание:

Метод `Initialize` выполняет инициализацию системы 1С:Предприятие.

Пример:

* здесь пример приводится на языке MS Visual Basic:

```
Dim v7 As Object
Set v7 = CreateObject("V77.Application")
result = v7.Initialize(v7.RMTrade, "/DC:\V7\DB /M", "")
```

EvalExpr

Вычислить выражение системы 1С:Предприятие.

Синтаксис:

EvalExpr(<СтрокаВыражения>)

Англоязычный синоним:

EvalExpr

Параметры:

<СтрокаВыражения>	Строковое выражение — выражение, записанное на встроенном языке 1С:Предприятие.

Возвращаемое значение:

Результат вычисленного выражения.

Описание:

Метод EvalExpr вычисляет выражение, записанное в параметре <СтрокаВыражения> на встроенном языке 1С:Предприятие и возвращает результат вычисления. Результатом выражения может быть число, строка, дата или значение любого агрегатного типа данных. Результат с неопределенным типом данных преобразуются к строковому типу.

Пример:

* здесь пример приводится на языке MS Visual Basic:

```
Dim v7 As Object
Dim Товары As Object
Set v7 = CreateObject("V77.Application")
result = v7.Initialize(v7.RMTrade, "/DC:\V7\DB /M", "")
Set Товары = v7.EvalExpr("ОтдатьСправочникТоваров()");
```

CreateObject

Создает объект агрегатного типа данных системы 1С:Предприятие и возвращает ссылку на него.

Синтаксис:

CreateObject(<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:

Метод CreateObject создает объект агрегатного типа данных системы 1С:Предприятие и возвращает ссылку на него. Данная функция обычно используется одновременно с неявным определением переменной и присвоением ей ссылки на объект агрегатного типа данных.

Пример:

```
Процедура Загрузить()
   Если ФС.Существуетфайл(Путь + "\NUL") = 0 Тогда
      Предупреждение("Путь информационной базы не найден!");
   Иначе
      V7 = СоздатьОбъект("V77.Application");
      Открыта = V7.Initialize(V7.RMTrade, "/d" + Путь +
                              " /M /N" + Пользователь, "");
      Если Открыта = 0 Тогда
         Предупреждение("Ошибка открытия информационной базы");
         Возврат;
      КонецЕсли;
      Импорт = V7.CreateObject("Справочник.Контрагенты");
      Импорт.ВыбратьЭлементы();
      Пока Импорт.ПолучитьЭлемент() = 1 Цикл
         Если Импорт.ЭтоГруппа() = 0 Тогда
            Сообщить(Импорт.Наименование);
         КонецЕсли;
      КонецЦикла;
   КонецЕсли;
КонецПроцедуры
```

ExecuteBatch

Выполнить последовательность операторов системы 1С:Предприятие.

Синтаксис:

ExecuteBatch(<СтрокаОператоров>)

Англоязычный синоним:

ExecuteBatch

Параметры:

<СтрокаОператоров>	Строковое выражение — текст программы на встроенном языке 1С:Предприятие.

Возвращаемое значение:

Значение логического типа: TRUE, если последовательность операторов выполнены успешно, или FALSE, если нет.

Замечание: В OLE Automation TRUE и FALSE имеют соответственно значения: -1 (минус единица) и 0.

Описание:

Метод ExecuteBatch выполняет последовательность операторов, записанную в параметре <СтрокаОператоров> на встроенном языке 1С:Предприятие.

Пример:

* здесь пример приводится на языке MS Visual Basic:

```
Sub Command1_Click ()
   Dim v7 As Object
   Dim Price As Object
   Set v7 = CreateObject("V77.Application")
   result = v7.Initialize(v7.RMTrade, "/DC:\V7\DB /M", "")
   Set Price = v7.EvalExpr("ОтдатьСправочникТоваров()")
   Price.New
   Price.Code = "112233"
   Price.Description = "Test string"
   Price.Write;
   BoolResult = v7.ExecuteBatch("ПроцедураРегл1();ПроцедураРегл2()")
End Sub
```

* Обращение к программе 1С:Предприятие из модуля MS Excel. В данном примере запускается и инициализируется конфигурация 1С:Предприятие с базой данных в каталоге C:\V7\DB в монопольном режиме. Далее в программе 1С:Предприятие создается объект типа "Справочник.Товары", где создается новая группа элементов с названием "***** Экспорт из Excel ******". Во вновь созданную группу каталога записываются данные из таблицы MS Excel. Здесь пример приводится на языке MS Visual Basic.

```
Sub Excel_to_trade()
   Dim trade As Object
   Dim Товар As Object
   Set trade = CreateObject("V77 .Application")
   result = trade.Iinitialize(trade.RMTrade, "/DC:\V7\DB /M", "")
   Set Товар = trade.EvalExpr("CreateObject(""Справочник.Товары"")")
   Товар.НоваяГруппа
   Товар.Наименование = "***** Экспорт из Excel ******"
   Товар.Записать
   Товар.ИспользоватьРодителя Товар.ТекущийЭлемент
   N = 100 'Количество строк в документе
   For Count = 1 To N Товар.Новый
      Товар.Наименование = Application.Cells(Count, 2).Value
      Товар.Розн_Цена = Application.Cells(Count, 3).Value
      Товар.Мел_Опт_Цена = Application.Cells(Count, 4).Value
      Товар.Опт_Цена = Application.Cells(Count, 5).Value
      Товар.Записать
   Next Count
End Sub
```

Работа системы 1С:Предприятие в качестве DDE сервера

Другим способом обращения к данным 1С:Предприятие а из внешних программ является режим DDE. Основное преимущество данного режима является динамическое обновление получаемых из 1С:Предприятие данных.

Система 1С:Предприятие является DDE сервером и предоставляет свой сервис в получении данных и уведомлении DDE клиента об их изменении.

В качестве DDE сервера 1С:Предприятие может возвращать значение некоторого выражения, записанного на встроенном языке 1С:Предприятие. Возвращаемое значение является строкой. Для того, чтобы рассматривать результат выражения в качестве числа или даты, например в ячейке MS Excel, может потребоваться задание формата значения.

Для доступа к 1С:Предприятие посредством DDE следует использовать DDE-объект с именем 1CV7|DDE.

Пример:

* Ниже приведены примеры размещения в ячейках MS Excel DDE связей с программой 1С:Предприятие. Первое выражение в примере обращается к стандартной функции встроенного языка 1С:Предприятие. Второе выражение в примере обращается к функции, записанной в глобальном модуле конфигурации программы 1С:Предприятие.

```
='1CV7'|DDE!'РабочаяДата()'
='1CV7'|DDE!'СуммаПоНарядам()'
```

Замечание 1: Для правильной интерпретации значения выражения в качестве числа или даты в MS Excel, нужно в операционной системе установить соответствующий формат представления даты и числа, как он принят в 1С:Предприятие.

Замечание 2: Не рекомендуется обращаться посредством DDE к функциям 1С:Предприятие, которые могут выполняться продолжительное время. Это связано с тем, что обращение к данной функции будет выполняться достаточно часто при обновлениях DDE связей.