Система стандартов и методик разработки конфигураций для платформы

1С:ПРЕДПРИЯТИЕ 8

Версия

январь 2020

Пустая страница

Оглавление

[Создание и изменение объектов метаданных 1](#_Toc31041292)

[Организация работы конфигурации 1](#_Toc31041293)

[Общие требования к конфигурации 1](#_Toc31041294)

[Имена объектов метаданных в конфигурациях 2](#_Toc31041295)

[Работа в разных часовых поясах 9](#_Toc31041296)

[Использование функциональных опций 12](#_Toc31041297)

[Использование параметров сеанса 15](#_Toc31041298)

[Использование подсистем 17](#_Toc31041299)

[Использование общих реквизитов 18](#_Toc31041300)

[Использование определяемых типов 18](#_Toc31041301)

[Правила создания общих модулей 19](#_Toc31041302)

[Работа с пользовательскими настройками 21](#_Toc31041303)

[Начальные действия при работе конфигурации 23](#_Toc31041304)

[Поддержка толстого клиента, управляемое приложение, клиент-сервер 24](#_Toc31041305)

[Технология разветвленной разработки конфигураций 25](#_Toc31041306)

[Особенности разработки конфигураций для ОС Linux и macOS 31](#_Toc31041307)

[Оформление карты маршрута бизнес-процесса 33](#_Toc31041308)

[Ограничения на переименование объектов метаданных 35](#_Toc31041309)

[Требования к установке и обновлению прикладных решений 37](#_Toc31041310)

[Несущественные предупреждения проверки конфигурации 38](#_Toc31041311)

[Учет версий конфигураций 38](#_Toc31041312)

[Общие сведения о выпуске конфигураций 38](#_Toc31041313)

[Нумерация редакций и версий 38](#_Toc31041314)

[Заполнение свойств конфигурации информацией о выпуске 39](#_Toc31041315)

[Организация хранения данных 40](#_Toc31041316)

[Общие сведения об организации хранения данных 40](#_Toc31041317)

[Уточнение сущности объекта метаданных 44](#_Toc31041318)

[Имя, синоним, комментарий 44](#_Toc31041319)

[Подсказка и проверка заполнения 46](#_Toc31041320)

[Использование кодов (номеров) объектов конфигурации 47](#_Toc31041321)

[Использование реквизитов строкового типа 49](#_Toc31041322)

[Ограничения на использование реквизитов составного типа 49](#_Toc31041323)

[Требования к проведению документов 50](#_Toc31041324)

[Использование активности движений 52](#_Toc31041325)

[Самодостаточность регистров 53](#_Toc31041326)

[Реквизит «Комментарий» у документов 53](#_Toc31041327)

[Удаление устаревших объектов метаданных из конфигурации 53](#_Toc31041328)

[Использование констант 54](#_Toc31041329)

[Работа с неактуальными (недействительными) объектами 55](#_Toc31041330)

[Использование предопределенных элементов 56](#_Toc31041331)

[Обработчики событий объектов 58](#_Toc31041332)

[Обработчик события ПередЗаписью 58](#_Toc31041333)

[Обработчик события ПриЗаписи 58](#_Toc31041334)

[Обработчик события ПередУдалением 58](#_Toc31041335)

[Обработчик события ПриКопировании 59](#_Toc31041336)

[Обработчик события ОбработкаПроверкиЗаполнения 59](#_Toc31041337)

[Обработчик события ОбработкаЗаполнения 61](#_Toc31041338)

[Обработчики событий ОбработкаПолученияПредставления и ОбработкаПолученияПолейПредставления 62](#_Toc31041339)

[Использование признака ОбменДанными.Загрузка в обработчиках событий объекта 63](#_Toc31041340)

[Регламентные задания 64](#_Toc31041341)

[Общие требования к регламентным заданиям 64](#_Toc31041342)

[Настройка расписания регламентных заданий 66](#_Toc31041343)

[Запуск регламентных заданий 67](#_Toc31041344)

[Ограничения на регламентные задания при работе в режиме сервиса 68](#_Toc31041345)

[Реализация обработки данных 70](#_Toc31041346)

[Работа с запросами 70](#_Toc31041347)

[Оформление текстов запросов 70](#_Toc31041348)

[Многократное выполнение однотипных запросов 73](#_Toc31041349)

[Проверка на пустой результат выполнения запроса 74](#_Toc31041350)

[Ограничение на использование конструкции "ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ" в запросах 74](#_Toc31041351)

[Использование ключевых слов "ОБЪЕДИНИТЬ" и "ОБЪЕДИНИТЬ ВСЕ" в запросах 76](#_Toc31041352)

[Упорядочивание результатов запроса 76](#_Toc31041353)

[Округление результатов арифметических операций в запросах 78](#_Toc31041354)

[Особенности использования в запросах оператора ПОДОБНО 79](#_Toc31041355)

[Псевдонимы источников данных в запросах 80](#_Toc31041356)

[Оптимизация запросов 81](#_Toc31041357)

[Общие требования по разработке оптимальных запросов 81](#_Toc31041358)

[Несоответствие индексов и условий запроса 81](#_Toc31041359)

[Разыменование ссылочных полей составного типа в языке запросов 84](#_Toc31041360)

[Ограничения на соединения с вложенными запросами и виртуальными таблицами 86](#_Toc31041361)

[Ограничения на использование вложенных запросов в условии соединения 88](#_Toc31041362)

[Обращения к виртуальным таблицам 89](#_Toc31041363)

[Эффективные условия запросов 90](#_Toc31041364)

[Разрешение итогов для периодических регистров сведений 94](#_Toc31041365)

[Эффективное обращение к виртуальной таблице «Остатки» 95](#_Toc31041366)

[Использование временных таблиц 96](#_Toc31041367)

[Обработка и модификация данных 96](#_Toc31041368)

[Транзакции: правила использования 96](#_Toc31041369)

[Использование управляемого режима блокировки 100](#_Toc31041370)

[Блокировка данных объекта для редактирования из кода 101](#_Toc31041371)

[Ответственное чтение данных 102](#_Toc31041372)

[Чтение отдельных реквизитов объекта из базы данных 105](#_Toc31041373)

[Запись событий в историю работы пользователя 106](#_Toc31041374)

[Избыточные блокировки и методы оптимизации 106](#_Toc31041375)

[Общие сведения об избыточных блокировках 106](#_Toc31041376)

[Сдвиг границы последовательности документов 107](#_Toc31041377)

[Режим разделения итогов для регистров бухгалтерии 108](#_Toc31041378)

[Режим разделения итогов для регистров накопления 110](#_Toc31041379)

[Блокирующее чтение остатков в начале транзакции 112](#_Toc31041380)

[Соглашения при написании кода 113](#_Toc31041381)

[Оформление модулей 113](#_Toc31041382)

[Тексты модулей 113](#_Toc31041383)

[Структура модуля 114](#_Toc31041384)

[Имена процедур и функций 120](#_Toc31041385)

[Описание процедур и функций 122](#_Toc31041386)

[Параметры процедур и функций 128](#_Toc31041387)

[Особенности использования структур в качестве параметров процедур и функций 130](#_Toc31041388)

[Правила образования имен переменных 131](#_Toc31041389)

[Работа с параметром «Отказ» в обработчиках событий 132](#_Toc31041390)

[Использование конструкций встроенного языка 132](#_Toc31041391)

[Общие требования к построению конструкций встроенного языка 132](#_Toc31041392)

[Перенос выражений 133](#_Toc31041393)

[Использование дублирующего кода 135](#_Toc31041394)

[Использование директив компиляции и инструкций препроцессора 137](#_Toc31041395)

[Определение типа значения переменной 139](#_Toc31041396)

[Получение метаданных объектов 139](#_Toc31041397)

[Обработчики событий модуля формы, подключаемые из кода 139](#_Toc31041398)

[Использование переменных в программных модулях 140](#_Toc31041399)

[Предварительная инициализация локальных переменных 142](#_Toc31041400)

[Использование Журнала регистрации 143](#_Toc31041401)

[Перехват исключений в коде 144](#_Toc31041402)

[Ограничение на использование оператора Перейти 148](#_Toc31041403)

[Использование прикладных объектов и универсальных коллекций значений 148](#_Toc31041404)

[Поиск в коллекциях значений 148](#_Toc31041405)

[Использование объекта РегистрСведенийМенеджерЗаписи 150](#_Toc31041406)

[Копирование строк между таблицами значений (табличными частями и т.п.) произвольной структуры 151](#_Toc31041407)

[Порядок записи движений документов 151](#_Toc31041408)

[Получение представлений для ссылочных значений в табличном документе 151](#_Toc31041409)

[Программное создание прикладных объектов 152](#_Toc31041410)

[Использование модуля объекта, модуля менеджера объекта и общих модулей 153](#_Toc31041411)

[Ограничения на использование экспортных процедур и функций 153](#_Toc31041412)

[Установка параметров выбора и связей параметров выбора для объектов метаданных 153](#_Toc31041413)

[Использование РеквизитФормыВЗначение и ДанныеФормыВЗначение 154](#_Toc31041414)

[Применение параметров отчета в СКД 154](#_Toc31041415)

[Использование объектов типа Структура 155](#_Toc31041416)

[Особенности сортировки в таблице значений 157](#_Toc31041417)

[Массовая конкатенация строк 157](#_Toc31041418)

[Клиент-серверное взаимодействие 158](#_Toc31041419)

[Общие вопросы безопасности 158](#_Toc31041420)

[Настройка прав доступа к данным 158](#_Toc31041421)

[Реализация обмена данными 158](#_Toc31041422)

[Разработка и использование библиотек 158](#_Toc31041423)

[Требования по локализации 158](#_Toc31041424)

[Проектирование интерфейсов для 8.3 158](#_Toc31041425)

[Разработка пользовательских интерфейсов 158](#_Toc31041426)

[Проектирование интерфейсов для 8.2 158](#_Toc31041427)

[Разработка пользовательских интерфейсов (обычное приложение) 158](#_Toc31041428)

# [Создание и изменение объектов метаданных](https://its.1c.ua/db/v8std/browse/13/-1/1)

## [Организация работы конфигурации](https://its.1c.ua/db/v8std/browse/13/-1/1/2)

### Общие требования к конфигурации

#std467

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Конфигурация должна использовать только штатные и документированные возможности платформы **1С:Предприятие**.

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1.2. Конфигурация должна быть одинаково рассчитана на работу со всеми СУБД, операционными системами, веб-браузерами и различными режимами работы, которые поддерживает платформа **1С:Предприятие**. В частности, в веб-клиенте все ключевые возможности конфигурации должны быть доступны пользователям без использования [расширения работы с файлами](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/900/i8100700.htm?_=1579516850), а [взаимодействие с пользователем должно быть организовано асинхронно](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100703.htm?_=1579516850). Дополнительные материалы:   * [Список поддерживаемых операционных систем и СУБД](http://v8.1c.ru/requirements/) * Особенности работы с различными СУБД см. в [приложении 8](http://its.1c.ru/db/v839doc#bookmark:dev:TI000001285) документации по платформе 1С:Предприятие 8.3 * Особенности режима низкой скорости соединения и работы веб-клиента см. в [приложении 7](http://its.1c.ru/db/v839doc#bookmark:dev:TI000001242) (там же) * [Особенности разработки конфигураций для ОС Linux](https://its.1c.ua/db/content/v8std/src/100/100/i8100723.htm?_=1579516850)   1.3. Конфигурация не должна содержать ошибок, обнаруживаемых при проверке конфигурации (конфигуратор – меню **Конфигурация** – **Проверка конфигурации…**). Кроме отдельных, обоснованных случаев:   * [Обработчики событий модуля формы, подключаемые из кода](https://its.1c.ua/db/content/v8std/src/300/200/i8100492.htm?_=1579516850); * [Ограничение на использование модальных окон и синхронных вызовов](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100703.htm?_=1579516850); * [Ограничение на установку признака «Вызов сервера»](https://its.1c.ua/db/content/v8std/src/500/i8100679.htm?_=1579516850); * [Несущественные предупреждения проверки конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100759.htm?_=1579516850).   1.4. Для поддержки обратной совместимости с различными собственными и сторонними решениями, внешними обработками и отчетами, разработанными на предыдущих версиях платформы **1С:Предприятие 8.0** и**8.1**, конфигурация также должна поддерживать запуск в режимах обычного приложения (толстый клиент) и внешнего соединения для администраторов (пользователей с полными правами). Для этого рекомендуется   * свойство конфигурации **«Использовать управляемые формы в обычном приложении»** установить в **Истина**, а свойство **«Использовать обычные формы в управляемом режиме»** – в **Ложь**. * придерживаться [общей схемы установки признаков общих модулей](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1579516850), * а саму разработку в Конфигураторе вести в режиме редактирования для обоих режимов запуска – управляемое и обычное приложение (меню **Сервис** – **Параметры** – закладка **Общие**).   Отказ от поддержки запуска конфигурации в режимах обычного приложения и внешнего соединения для администраторов возможен только в отдельных, обоснованных случаях. |

1.5. При проектировании тех или иных технических решений, при разработке пользовательского интерфейса, отчетов и т.п. не рекомендуется отходить от умолчаний платформы **1С:Предприятие**. Реализация альтернативных вариантов технических решений допустима только в отдельных, обоснованных случаях.

2.1. Имена, синонимы, комментарии объектов метаданных, общих модулей, а также любая текстовая информация (которая выводится пользователю или предназначена для разработчика/внедренца) должны быть составлены по правилам русского языка и, в частности, не должны содержать грамматических ошибок.

2.2. В конфигурации не должно быть неиспользуемых объектов метаданных (справочников, документов, разделов командного интерфейса и т.п.) и программного кода (общих модулей, процедур, функций, переменных и т.п.), который не используется ни в самой конфигурации, ни для интеграции с другими системами.

2.3. Объекты метаданных верхнего уровня, такие как **Справочники**, **Документы, Общие модули** и т.д. рекомендуется сортировать в дереве метаданных по имени. Подчиненные объекты метаданных, такие как реквизиты, измерения, формы, располагаются в дереве метаданных в соответствии с проектной логикой.

Исключение составляют:

* [общие реквизиты](https://its.1c.ua/db/content/v8std/src/100/100/i8100677.htm?_=1579516850) (т.к. для общих реквизитов, являющихся разделителями, порядок следования в дереве метаданных должен подбираться, исходя из требуемого порядка установки параметров сеанса).
* [объекты с префиксом "Удалить"](https://its.1c.ua/db/content/v8std/src/100/300/i8100534.htm?_=1579516850) (англ. "Obsolete"), которые допустимо размещать в конце соответствующей ветки метаданных;

### Имена объектов метаданных в конфигурациях

#std550

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

См. также: [*общие правила наименования метаданных*](https://its.1c.ua/db/content/v8std/src/100/300/i8100474.htm?_=1579516850)

| **п/п** | **Объекты метаданных** | **Правила наименования** | **Область применения (уточнение)** |
| --- | --- | --- | --- |
| 1. | Подсистемы | Согласно общим правилам наименования метаданных. Например: **Финансы**, **Маркетинг**, **НастройкаИАдминистирование**.  См. также: [Использование подсистем](https://its.1c.ua/db/content/v8std/src/100/100/i8100543.htm?_=1579516850) | Управляемое приложение Обычное приложение |
| 2. | Общие модули | См. [Правила создания общих модулей](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1579516850) |  |
| 3. | Параметры сеанса | Согласно общим правилам наименования метаданных. Например: **ТекущийПользователь**, **ОбменДаннымиВключен**, **РаботаСВнешнимиРесурсамиЗаблокирована**.  См. также: [Использование параметров сеанса](https://its.1c.ua/db/content/v8std/src/100/100/i8100413.htm?_=1579516850) |  |
| 4. | Роли | При именовании ролей рекомендуется придерживаться двух схем:  «прикладные» роли, соответствующие должностным обязанностям определенной категории пользователей информационной системы, следует именовать от названия должности, например **Бухгалтер**, **Кассир**, **Администратор**.  роли, дающие доступ к более «мелким» функциональным блокам для более «тонкой» настройки прав доступа пользователей, рекомендуется именовать от описания разрешаемого действия. Например: **ДобавлениеИзменениеНСИ**, **ЧтениеДополнительныхСведений**, **ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок**.  См. также: [Стандартные роли](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1579516850) |  |
| 5. | Общие реквизиты | Согласно общим правилам именования метаданных. | Управляемое приложение Обычное приложение |
| 6. | Планы обмена | Имена планов обмена рекомендуется называть по следующим принципам:   * в именах планов обмена РИБ (признак **Распределенная ИБ** включен) кратко описываются правила синхронизации данных. Например: **Полный**, **ПоОрганизациям**, **ПоСкладамИОрганизациям** * имена планов обмена между различными конфигурациями следует формировать из имени источника и имени приемника. Имена планов обмена в источнике и приемнике должны быть одинаковыми. Например: **ОбменУправлениеНебольшойФирмойБухгалтерияПредприятия**, **ОбменУправлениеТорговлейРозничнаяТорговля**.   При необходимости организации обмена с разными версиями (редакциями) конфигураций, к именам приемника и источника добавляются номера версий (редакций). Например, **ОбменУправлениеТорговлей110РозничнаяТорговля10**(обмен данными между конфигурациями редакций 11.0 и 1.0) |  |
| 7. | Критерии отбора | Имена критериев отбора рекомендуется задавать во множественном числе, образуя имя от названия списка объектов, которые он отбирает. Например: **СвязанныеДокументы**, **ФайлыВТоме**. | Управляемое приложение Обычное приложение |
| 8. | Подписки на события | Имена подписок на события рекомендуется задавать от сути выполняемого действия и образовывать от неопределенной формы глагола. Например, неправильно **ЗапретРедактированияРеквизитовОбъектовПередЗаписьюОбъекта НастройкаПорядкаЭлементовПередЗаписью**  правильно: **ПроверитьИзменениеРеквизитовОбъекта ПересчитатьПорядковыйНомерЭлемента** |  |
| 9. | Регламентные задания | Имена регламентных заданий рекомендуется давать в единственном числе и образовывать от существительного. Например, неправильно **УстановитьПериодРасчитанныхИтогов** **УведомитьИсполнителейОНовыхЗадачах**  правильно **УстановкаПериодаРасчитанныхИтогов** **УведомлениеИсполнителейОНовыхЗадачах** | Управляемое приложение Обычное приложение |
| 10. | Функциональные опции | Имена функциональных опций, связанных с константами, рекомендуется образовывать от описания включаемой (или выключаемой) с их помощью функциональности. Например, для функциональных опций типа **Булево**: **ИспользоватьБизнесПроцессыИЗадачи** **ИспользоватьВерсионированиеОбъектов**  для функциональной опции других типов: **ПрефиксИнформационнойБазы** (тип **Строка**) **ВариантыВерсионированияОбъектов** (параметризуемая функциональная опция, связанная с регистром сведений)  См. также: [Использование функциональных опций](https://its.1c.ua/db/content/v8std/src/100/100/i8100470.htm?_=1579516850) |  |
| 11. | Параметры функциональных опций | Имена параметров функциональных опций рекомендуется задавать от описания параметра. При этом необязательно, чтобы имя параметра функциональной опции совпадало с наименование реквизитов объектов, на которые ссылается параметр. Например:  **Организация** – связан со справочником **Организации**; **ТипОбъектаКонфигурации** – связан с двумя ресурсами регистров сведений:   * **РегистрСведений.НазначениеДополнительныхОбработок.Измерение.ТипОбъекта**, и * **РегистрСведений.НастройкаВерсионированияОбъектов.Измерение.ТипОбъекта**   См. также: [Использование функциональных опций](https://its.1c.ua/db/content/v8std/src/100/100/i8100470.htm?_=1579516850) |  |
| 12. | Определяемые типы | Имена определяемых типов рекомендуется задавать в единственном числе и образовывать от их назначения. При этом не следует называть их так же, как называются другие типы данных (например: «Строка», «Число», …), и не использовать слова, от удаления которых смысл не меняется (например: «Тип...», «Объект…», «Ссылка…»).  Например, неправильно: **Строка25**, **СсылкиНаКонтактыВзаимодействий**  Правильно: **АртикулНоменклатуры** – строка фиксированной длины 25 символов, которая используется в справочнике номенклатуры организации, справочниках номенклатуры поставщиков, отчетов и обработках, предназначенных для работы с номенклатурой. **КонтактВзаимодействий** – составной тип, включающий в себя ссылки на различные справочники, элементы которых являются контактами взаимодействий (электронных писем, телефонных звонков, встреч и пр.). Например, **Пользователи**, **КонтактныеЛицаПартнеров**, **Партнеры** и другие.  См. также: [Использование определяемых типов](https://its.1c.ua/db/content/v8std/src/100/100/i8100704.htm?_=1579516850) | Управляемое приложение Обычное приложение |
| 13. | Хранилища настроек | Согласно общим правилам наименования метаданных. Например: **ХранилищеВариантовОтчетов**. | Управляемое приложение Обычное приложение |
| 14. | Общие формы | Имена общих форм рекомендуется образовывать от существительных. При этом следует избегать в имени форм слов, от удаления которых смысл не меняется, например: «Форма…», «Окно…», «Диалог…».  Примеры: **НастройкаСистемы**, **МоиНастройки**, **ПараметрыПроксиСервера**,**ВыборОбъектовМетаданных**. |  |
| 15. | Общие команды | Имена общих команд рекомендуется задавать по следующим принципам:   * если команда предназначена для размещения в панели навигации той или иной формы, или раздела командного интерфейса, то ее имя должно обозначать список объектов, которые она открывает. Например: **ДополнительныеОтчетыИОбработкиЗаполнениеОбъекта**, **ЗадачиПоБизнесПроцессу** * в остальных случаях, как правило, имена общих команд образуются от неопределенной формы глагола, обозначающего действие команды, например: **ВыполнитьСопоставление**, **УстановитьРасширениеРаботыСФайлами** |  |
| 16. | Группы команд | Имена групп команд рекомендуется образовывать от существительных. Например, **ПараметрыОбменаДанными**, **Печать**. |  |
| 17. | Интерфейсы | Согласно общим правилам наименования метаданных.  См. также: [Общие правила построения интерфейсов](https://its.1c.ua/db/content/v8std/src/1%C2%A0300/i8100500.htm?_=1579516850), [Общие интерфейсы](https://its.1c.ua/db/content/v8std/src/1%C2%A0300/i8100501.htm?_=1579516850) | Обычное приложение |
| 18. | Общие макеты | Имена общих макетов рекомендуется образовывать от существительных, дающих краткое представление о содержимом или назначении макета. При этом следует избегать в имени слов, от удаления которых смысл не меняется, например: «Макет…». Примеры: **ДополнительнаяОбработка**, **КомпонентаTWAIN**, **ОписаниеИзмененийСистемы**. |  |
| 19. | Общие картинки | Согласно общим правилам наименования имен метаданных. Например: **Найти** – универсальная картинка для команд поиска, для использования в различных подсистемах конфигурации. **ЗакрепитьВариантОтчета** – картинка команды «Закрепить вариант отчета».  Допускается указывать спецификаторы размера, например: **Папка** – картинка размером 16x16 пикселей **УправлениеПоиском32** – картинка размером 32x32 пикселей **ДлительнаяОперация48** – картинка размером 48x48 пикселей  Для обозначения картинок-коллекций к имени добавляется префикс **Коллекция**. Например: **КоллекцияВариантыВажностиЗадачи**.  При этом следует избегать в имени общих картинок слов, от удаления которых смысл не меняется, например: «Картинка…», «Изображение…», «Пиктограмма…». |  |
| 20. | XDTO-пакеты | Имена XDTO-пакетов рекомендуется образовывать на русском языке от существительных, дающих краткое представление о содержимом или назначении пакета. При этом следует избегать в имени слов, от удаления которых смысл не меняется, например: «Пакет…», «ХDTO…».  Пример: **Файлы**. | Управляемое приложение Обычное приложение |
| 21. | Web-сервисы | Имена Web-сервисов рекомендуется образовывать на английском языке от существительных, дающих краткое представление об их назначении. Не рекомендуется использовать кириллицу, так как сторонние информационные системы могут ее не поддерживать, а также слова, от удаления которых смысл не меняется, например: «Service», «WebService».  Примеры: **Files**, **Accounts**, **FlightStatus**.  Имена операций Web-сервисов, а также их параметры рекомендуется также писать на английском языке.  Пример: **GetCurrencyRate** | Управляемое приложение Обычное приложение |
| 22. | WS-ссылки | Имена WS-ссылок рекомендуется образовывать от существительных, дающих краткое представление о назначении Web-сервиса. При этом следует избегать в имени слов, от удаления которых смысл не меняется, например: «WebСервис…», «Сервис…», «Ссылка…».  Примеры: **ДанныеОтгрузки**, **КонверторВалют**. |  |
| 23. | Элементы стиля | Имена элементов стиля рекомендуется образовывать от существительных, дающих краткое представление об их назначении. Например: **ВыполненнаяЗадача**, **ПоясняющийТекст**, **НеСтартованныйБизнесПроцесс**.  Также в имени элемента стиля допускается уточнение по поводу определяемого параметра стиля, например: **УдаленныйДополнительныйРеквизитЦвет**, **УдаленныйДополнительныйРеквизитШрифт**.  См. также: [Элементы стиля](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/i8100667.htm?_=1579516850) (для режима обычного приложения см. [Стили](https://its.1c.ua/db/content/v8std/src/1%C2%A0300/i8100524.htm?_=1579516850)) | Управляемое приложение Обычное приложение |
| 24. | Стили | Согласно общим правилам наименования метаданных.  См. также: [Стили](https://its.1c.ua/db/content/v8std/src/1%C2%A0300/i8100524.htm?_=1579516850) (для обычного приложения) | Обычное приложение |
| 25. | Языки | Имя следует образовывать от наименования языка пользовательского интерфейса программы: **Русский**, **Английский**, и т.п. |  |
| 26. | Константы | Имена констант рекомендуется задавать по следующим принципам:   * Если константа связана с функциональной опцией, то созвучно функциональной опции; * В остальных случаях имя константы образуется от описания объекта, значение которого она хранит. Например: **ТипХраненияФайлов**, **НастройкаПроксиСервера**. * Если тип значения константы – **Булево**, то имя может быть образовано от неопределенной формы глагола, обозначающего включаемое или выключаемое действие. Например: **ВыполнятьПроверкуЭЦПНаСервере**, **ИзвлекатьТекстыФайловНаСервере**, **ИзменятьЗаданияЗаднимЧислом**.   При этом следует избегать в имени констант слов, от удаления которых смысл не меняется, например: «Константа…». |  |
| 27. | Справочники | Имена справочников рекомендуется давать во множественном числе и образовывать от описания списка объектов, значения которых хранятся в справочнике. Например: **Валюты**, **ГруппыИсполнителейЗадач**, **ПрофилиГруппДоступа**, **Пользователи.**  При этом следует избегать в именах справочников слов, от удаления которых смысл не меняется, например: «Справочник…». |  |
| 28. | Документы | Имена документов, напротив, даются в единственном числе. Например: **ЗаказПокупателя**, **ПеремещениеТоваров**, **Анкета**.  При этом следует избегать в именах документов слов, от удаления которых смысл не меняется, например: «Документ…».  При выборе имени документа следует различать два случая:  1. В первую очередь, следует стараться отразить в имени документа суть процесса, который отражается в системе этим документом. При этом само имя должно быть максимально лаконичным, рекомендуется избегать слов «Накладная…», «Акт…» и т.п.  Например, в системе автоматизирован процесс «Сверка взаиморасчетов», который завершается подписанием сторонами, участвующими в сверке, печатного документа «Акт сверки  товаров».  Поскольку в данном случае в системе документом фиксируется именно процесс, то документ называется **СверкаВзаиморасчетов**.  2. Если документ не отражает какой-либо процесс в системе, а предназначен только для получения соответствующей печатной формы, то допустимо образовывать имя документа от имени печатной формы. В этом случае допустимо использовать слова «Накладная», «Акт» и т.п. в имени документа. Как правило, у такого документа нет статусов, по нему не вводятся на основании другие документы, а сам процесс получения печатной формы может быть автоматизирован другими документами.   Например, для получения печатной формы «Товарно транспортная накладная» (ТТН) в системе имеется документ, который содержит реквизиты, специфичные для данного печатного документа. При этом поскольку весь процесс формирования ТТН связан с другими документами («Реализация  товаров и  услуг», «Перемещение товаров»), то документ целесообразно назвать от имени печатной формы: **ТоварноТранспортнаяНакладная**. |  |
| 29. | Журналы документов | Имена журналов документов рекомендуется задавать во множественном числе и образовывать от описания списка объектов, которые содержатся в журнале. Например: **СкладскиеДокументы**, **КорректировкиНДС**.  При этом следует избегать в именах слов, от удаления которых смысл не меняется, например: «Документы…». |  |
| 30. | Перечисления | Имена перечислений в конфигурации рекомендуется задавать во множественном числе.  Например, неправильно **ДействиеСДокументамиПоДвойномуЩелчку** **ВажностьЗадачи** **SMTPАутентификация**  Правильно: **ДействияСДокументамиПоДвойномуЩелчку** **ВариантыВажностиЗадачи** **ВидыSMTPАутентификации**  Любое исключение из этого правила должно быть обоснованным. Например: **ПолФизическогоЛица** |  |
| 31. | Отчеты | 1. Имена отчетов и вариантов отчетов рекомендуется образовывать от имени существительного. Например: **ДинамикаИзмененийФайлов**, **СписокЗависшихЗадач**, **СправкаОбИсполнительскойДисциплине**.  2. Рекомендуется предусматривать вывод заголовка, если отчет или вариант отчета предназначен для печати.   |  | | --- | | Методическая рекомендация (полезный совет)  Для отчетов с макетом заголовок должен располагаться в самом макете. Для вариантов отчетов без макета достаточно установить свойство **"Заголовок"** на закладке **"Дополнительные настройки".** |   3. При выборе представления варианта отчета следует придерживаться следующих рекомендаций:   * Представление должно лаконично и однозначно описывать суть данных, которые выводятся в этом варианте отчета.  Например, для отчета «Валовая прибыль», неправильно: «Основной», «По заказам» (подобные названия могут встречаться у вариантов других отчетов)  Правильно: «Валовая прибыль по контрагентам», «Валовая прибыль по заказам» * Если для варианта отчета выводится заголовок, то его представление должно совпадать с заголовком печатной формы. Это позволит пользователю однозначно определять вариант отчета по его печатной форме. * Не рекомендуется образовывать представление от имени отчета с уточнением специфики варианта отчета в скобках или с использованием других разделителей. Это может перегрузить списки вариантов отчетов повторениями, разделителями и уточнениями, снизив простоту визуального восприятия.   4. При этом следует избегать в именах отчетов и вариантов отчетов слов, от удаления которых смысл не меняется, например: «Отчет…». | Управляемое приложение Обычное приложение |
| 32. | Обработки | Имена обработок рекомендуется образовывать от имени существительного. Например: **КонтрольЖурналаРегистрации**, **РегламентныеИФоновыеЗадания**, **УправлениеИтогамиИАгрегатами**.  При этом следует избегать в именах обработок слов, от удаления которых смысл не меняется, например: «Обработка…».  Если форму обработки предполагается открывать из панели навигации той или иной формы, или раздела командного интерфейса, то имя обработки должно совпадать с именем команды для ее открытия. Например, команда **РегламентныеИФоновыеЗадания** открывает обработку **РегламентныеИФоновыеЗадания**. |  |
| 33. | Планы видов характеристик | Имена планов видов характеристики рекомендуется задавать во множественном числе и образовывать от описания списка объектов, которые перечисляются в плане видов характеристик. Например: **ВидыДоступа**, **ВопросыДляАнкетирования**, **ДополнительныеРеквизитыИСведения** | Управляемое приложение Обычное приложение |
| 34. | Планы счетов | Имена планов счетов рекомендуется задавать в единственном числе, образуя имя от существительного, дающего краткое представление о назначении плана счетов. При этом следует избегать в имени плана счетов слов, от удаления которых смысл не меняется, например: «ПланСчетов…». В то же время, в синониме может задаваться полное наименование. Например (имя – синоним):   **ЕПСБУ** - "ЕПСБУ"   **БухгалтерскийУчет**  - "План счетов бухгалтерского учета"   **УправленческийУчет** - "План счетов управленческого учета"   **МеждународныйУчет**  - "План счетов международного учета"   **Бюджетирование**     - "План счетов для бюджетирования"   и т.п.   Для дополнительных уточнений можно использовать свойство **Пояснение**, значение которого выводится в подсказке к команде открытия плана счетов. Например:   * **НалоговыйУчет** - синоним "План счетов налогового учета", пояснение: "План счетов налогового учета (по налогу на прибыль)" * **ЕПСБУ** - синоним "ЕПСБУ", пояснение: "Единый план счетов бюджетного учета" | Управляемое приложение Обычное приложение |
| 35. | Планы видов расчета | Имена планов видов расчета рекомендуется задавать во множественном числе и образовывать от описания списка объектов, которые перечисляются в плане видов расчета. Например: **ОсновныеНачисления**, **УправленческиеНачисления**, **Удержания**. | Управляемое приложение Обычное приложение |
| 36. | Регистры сведений, регистры накопления | Имена регистров сведений, регистров накопления рекомендуется задавать во множественном числе и образовывать от описания списка записей, которые содержатся в регистре. Например: **ДокументыФизическихЛиц**, **ФайлыВРабочемКаталоге**, **ДвиженияДенежныхСредств**. |  |
| 37. | Регистры бухгалтерии, регистры расчета | Имена регистров бухгалтерии и регистров расчета рекомендуется образовывать от описания списка записей, которые содержатся в регистре. Например: **Хозрасчетный**, **Начисления**. | Управляемое приложение Обычное приложение |
| 38. | Бизнес-процессы | Имена бизнес-процессов рекомендуется задавать как и имена документов, в единственном числе. Например: **Задание**, **Согласование**, **Утверждение**, **Поручение**. | Управляемое приложение Обычное приложение |
| 39. | Задачи | Имена задач бизнес-процессов рекомендуется задавать в единственном числе. Например: **ЗадачаИсполнителя**. | Управляемое приложение Обычное приложение |
| 40. | Внешние источники данных | Имена внешних источников данных рекомендуется образовывать от описания импортируемых данных. При этом следует избегать в имени слов, от удаления которых смысл не меняется: «Данные…», «ИсточникДанных…».  Примеры: **ДоговорыУправленческойУчетнойСистемы**, **ФайлыОсновнойСЭД**.  Таблицы внешних источников данных рекомендуется называть согласно общим правилам наименования объектов метаданных.  Примеры: **Договоры**, **Номенклатура**. | Управляемое приложение Обычное приложение |

См. также

* [Ограничения на переименование объектов метаданных](https://its.1c.ua/db/content/v8std/src/100/100/i8100706.htm?_=1579516850)

### Работа в разных часовых поясах

#std643

Область применения: управляемое приложение, обычное приложение.

1. Конфигурации должны быть рассчитаны на работу в условиях, когда часовой пояс на серверном компьютере не совпадает с реальным часовым поясом пользователей информационной базы. Например, с сервером, расположенным в Москве, работают сотрудники компании из Владивостока, и при этом все операции в системе должны выполняться по местному времени (Владивостока).

Такой сценарий работы часто востребован в клиент-серверных информационных базах и в прикладных решениях в модели сервиса (SaaS).

2.1. Во всех серверных процедурах и функциях вместо функции **ТекущаяДата**, которая возвращает дату и время серверного компьютера, следует использовать функцию **ТекущаяДатаСеанса**, которая приводит время сервера к часовому поясу пользовательского сеанса.

2.2. В тех случаях, когда требуется «универсальная» отметка времени, не зависящая от часового пояса текущего сеанса пользователя, в контексте которого выполняется серверный вызов, следует использовать функцию **УниверсальноеВремя**. Например, для определения момента перезаполнения закешированных данных, для получения времени последнего выполнения фонового задания и т.п.

2.3. При использовании методов платформы, возвращающих локальную дату серверного компьютера, следует приводить ее либо к универсальному времени, либо к времени пользовательского сеанса. Например:

ДатаАктуальностиУниверсальная = УниверсальноеВремя(ПолнотекстовыйПоиск.ДатаАктуальности());  
ДатаАктуальности = МестноеВремя(ДатаАктуальностиУниверсальная, ЧасовойПоясСеанса());

3.1. В клиентском коде использование функции **ТекущаяДата** также недопустимо. Это требование обусловлено тем, что текущее время, вычисленное в клиентском и серверном коде, не должно различаться.

Например, с сервером, расположенным в Москве, работают пользователи из Киева. Функция **ТекущаяДата** для клиентского компьютера возвращает 10:00, а для сервера – 11:00. В то же время, функция **ТекущаяДатаСеанса** вернет на сервере 10:00, если в информационной базе установлено киевское время (с помощью метода **УстановитьЧасовойПоясИнформационнойБазы**).

Как правило, вместо вызова функции **ТекущаяДата** на клиенте необходимо

* передавать с сервера на клиент время и дату, приведенную к часовому поясу пользовательского сеанса;
* при работе с документами на клиенте, использовать дату документа.

Рассмотрим типовые случаи на примерах.

3.2. В алгоритме закрытия месяца с клиента на сервер передается дата, по которой далее определяется, какой месяц будет закрываться.

**Неправильно:**

&НаКлиенте  
Процедура КомандаОткрытьЗакрытиеМесяца(Команда)  
 ТекущиеДанные = Элементы.Список.ТекущиеДанные;  
 Если ТекущиеДанные = Неопределено Тогда  
  ТекДата  = ТекущаяДата();  // вызов ТекущаяДата() на клиенте  
 Иначе  
  ТекДата  = ТекущиеДанные.Дата;  
 КонецЕсли;  
 ПараметрыФормы = Новый Структура;  
 ПараметрыФормы.Вставить("ПериодРегистрации", ТекДата);  
 ОткрытьФорму("Обработка.ЗакрытиеМесяца.Форма.Форма", ПараметрыФормы, ЭтотОбъект);  
…

а затем в форме обработки:

&НаСервере  
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)  
 ЗаполнитьЗначенияСвойств(Объект, Параметры);  
 Если Не ЗначениеЗаполнено(Объект.ПериодРегистрации) Тогда  
  Объект.ПериодРегистрации = НачалоМесяца(ТекущаяДата());  
 КонецЕсли;  
…

**Правильно**

перенести получение текущей даты на сервер:

&НаКлиенте  
Процедура КомандаОткрытьЗакрытиеМесяца(Команда)  
 ТекущиеДанные = Элементы.Список.ТекущиеДанные;  
 Если ТекущиеДанные = Неопределено Тогда  
  ТекДата  = Неопределено; // нет вызова ТекущаяДата() на клиенте  
 Иначе  
  ТекДата  = ТекущиеДанные.Дата;  
 КонецЕсли;  
 ПараметрыФормы = Новый Структура;  
 ПараметрыФормы.Вставить("ПериодРегистрации", ТекДата);  
 ОткрытьФорму("Обработка.ЗакрытиеМесяца.Форма.Форма", ПараметрыФормы, ЭтотОбъект);  
…

и в форме обработки использовать для этого функцию **ТекущаяДатаСеанса**:

&НаСервере  
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)  
 ЗаполнитьЗначенияСвойств(Объект, Параметры);  
 Если Не ЗначениеЗаполнено(Объект.ПериодРегистрации) Тогда  
  Объект.ПериодРегистрации = НачалоМесяца(ТекущаяДатаСеанса());  
 КонецЕсли;  
…

3.3. При работе с документами следует рассмотреть возможность использования даты самого документа вместо текущей даты. Например, в реализации подбора номенклатуры в табличную часть документа, в форму подбора из клиентского кода передается дата расчетов для вывода цен и остатков на эту дату.

**Неправильно**:

&НаКлиенте  
Процедура ПодборТовары(Команда)  
 ПараметрыПодбора = Новый Структура;  
 ДатаРасчетов = ?(НачалоДня(Объект.Дата) = НачалоДня(ТекущаяДата()),  
  Неопределено, Объект.Дата); // вызов ТекущаяДата() на клиенте  
 ПараметрыПодбора.Вставить("ДатаРасчетов", ДатаРасчетов);  
 ...  
 ОткрытьФорму("Обработка.ПодборНоменклатуры.Форма.Форма", ПараметрыПодбора,  
  ЭтотОбъект, УникальныйИдентификатор);  
…

**Правильно**

передавать на сервер дату документа, а вычисление даты расчетов выполнять на сервере:

&НаКлиенте  
Процедура ПодборТовары(Команда)  
 ПараметрыПодбора = Новый Структура;  
 ПараметрыПодбора.Вставить("ДатаРасчетов", Объект.Дата);  
 ...  
 ОткрытьФорму("Обработка.ПодборНоменклатуры.Форма.Форма", ПараметрыПодбора,  
  ЭтотОбъект, УникальныйИдентификатор);

Другой пример. При подборе документов для зачета аванса в форме подбора устанавливается отбор по условию «дата документов аванса не больше переданной в форму».

**Неправильно**:

&НаКлиенте  
Процедура ЗачетАвансовДокументАвансаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)  
 СтандартнаяОбработка = Ложь;  
 ПараметрыФормы = Новый Структура;  
  ПараметрыФормы.Вставить("КонецПериода",                                     
?(ЗначениеЗаполнено(Параметры.Ключ), Объект.Дата - 1, КонецДня(ТекущаяДата()))); // вызов ТекущаяДата() на клиенте  
 ...  
  ОткрытьФорму("Документ.ДокументРасчетовСКонтрагентом.ФормаВыбора",          
ПараметрыФормы, Элемент);  
...

**Правильно**

вычислять параметр КонецПериода по дате документа:

&НаКлиенте  
Процедура ЗачетАвансовДокументАвансаНачалоВыбора(Элемент, ДанныеВыбора,  
СтандартнаяОбработка)  
 СтандартнаяОбработка = Ложь;  
 ПараметрыФормы = Новый Структура;  
  ПараметрыФормы.Вставить("КонецПериода", ?(ЗначениеЗаполнено(Параметры.Ключ),  
  Объект.Дата - 1, КонецДня(Объект.Дата)));  
 ...  
  ОткрытьФорму("Документ.ДокументРасчетовСКонтрагентом.ФормаВыбора", ПараметрыФормы, Элемент);

3.4. В остальных случаях при использовании Библиотеки стандартных подсистем рекомендуется использовать функцию **ДатаСеанса** общего модуля **ОбщегоНазначенияКлиент**.

4. Исключения из правил 2 и 3 возможны в отдельных, обоснованных случаях, когда требуется использовать действительно текущее время серверного компьютера. Такие исключения должны быть обоснованы в тексте комментария к вызову.

5. Следует избегать в коде одной процедуры (функции) многократного обращения к функции **ТекущаяДатаСеанса** (**ТекущаяДата**), так как возвращаемые значения будут отличаться друг от друга.

**Неправильно**

ДатаПоследнегоОповещения = ТекущаяДатаСеанса();  
ДатаСледующегоОповещения = РассчитатьДату() + ТекущаяДатаСеанса();

**Правильно**

использовать ранее рассчитанные дату и время:

ДатаПоследнегоОповещения = ТекущаяДатаСеанса();  
ДатаСледующегоОповещения = РассчитатьДату() + ДатаПоследнегоОповещения;

### Использование функциональных опций

#std470

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. В случае если некоторая функциональность конфигурации является необязательной для использования, то для управления доступностью такой функциональности на стадии внедрения следует применять функциональные опции. Для хранения значений функциональных опций в информационной базе необходимо завести в конфигурации соответствующие данные (например, константы).

Допустим, в конфигурации есть функциональность версионирования данных информационной базы, которая является необязательной. Для управления доступностью этой функциональности необходимо:

* создать функциональную опцию **ИспользоватьВерсионированиеОбъектов**, которая определяет использование прикладного механизма конфигурации для текущей информационной базы
* создать константу **ИспользоватьВерсионированиеОбъектов** типа **Булево** для хранения значения этой функциональной опции
* в свойстве **Хранение** функциональной опции указать константу **ИспользоватьВерсионированиеОбъектов**.

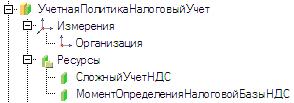
После этого, те или иные объекты конфигурации можно «привязать» к функциональной опции, включив их в ее состав, а в случае необходимости управления доступностью кода – использовать метод **ПолучитьФункциональнуюОпцию**:

ИспользуетсяМеханизмВерсионирования = ПолучитьФункциональнуюОпцию("**ИспользоватьВерсионированиеОбъектов**");

Таким образом, набор функциональных опций описывает функциональность конфигурации, доступность которой на этапе внедрения можно настроить в зависимости от требований конкретного предприятия. При этом платформа автоматически изменяет пользовательский интерфейс в соответствии с установленными значениями функциональных опций.

Функциональные опции могут также влиять на бизнес-логику. Для чего применяются функциональные опции не только булева типа, но и других типов, например, ссылки на справочники или значения перечислений.

1.2. Доступность функциональности может задаваться не только для информационной базы в целом, но и в зависимости от контекста применения этой функциональности. Допустим, в конфигурации необходимо управлять применением функциональности сложного учета НДС, но не в целом для всей информационной базы, а в зависимости от организации. Для этого необходимо:

* создать функциональную опцию **УчетнаяПолитикаСложныйУчетНДС**
* создать параметр функциональной опции **Организация**, поскольку значение функциональной опции зависит от организации (если такого параметра в конфигурации еще нет)
* создать регистр сведений **УчетнаяПолитикаНалоговыйУчет** для хранения значений этой функциональной опции, с измерением **Организация** и ресурсами, которые необходимы для управления функциональностью учета НДС  
    
  
* в свойстве **Хранение** функциональной опции указать ресурс регистра сведений **СложныйУчетНДС**
* для параметра функциональной опции **Организация** указать в свойстве **Использование** измерения **Организация** регистра сведений **УчетнаяПолитикаНалоговыйУчет**.

После этого, для того чтобы в той или иной форме значение функциональной опции соответствовало контексту, необходимо устанавливать значения параметров функциональной опции, например, так:

УстановитьПараметрыФункциональныхОпцийФормы(Новый Структура("Организация", <ТребуемаяОрганизация>));

В случае необходимости управления доступностью кода в зависимости от значения такой функциональной опции, ее значение можно получать, например, так:

ПараметрыУчетнойПолитики = Новый Структура("УчетнаяПолитикаОрганизация", <ТребуемаяОрганизация>);  
СложныйУчетНДС = ПолучитьФункциональнуюОпцию("УчетнаяПолитикаСложныйУчетНДС", ПараметрыУчетнойПолитики);  
МоментОпределенияНалоговойБазыНДС = ПолучитьФункциональнуюОпцию("УчетнаяПолитикаМоментОпределенияНалоговойБазыНДС ", ПараметрыУчетнойПолитики);

**Внимание:** следует учитывать, что описанный здесь вариант применения функциональных опций не является единственным вариантом их использования.  
Подробнее см. в документации по платформе **1С:Предприятие**.

1.3. Не следует использовать функциональные опции не по назначению, например:

* создавать функциональные опции ради управления видимостью элементов управления конкретной формы. С помощью функциональных опций следует управлять доступностью той или иной функциональности для всей конфигурации (и, как следствие, доступностью элементов форм и команд во всей конфигурации, а не в одной отдельно взятой форме);
* использовать функциональные опции для оптимизации доступа к тем или иным данным информационной базы (хранения значений на сервере 1С:Предприятия). Для этой цели предназначены [модули с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100724.htm?_=1579516850).

#### Установка и получение значений функциональных опций

2.1 Платформа **1С:Предприятие** не предоставляет каких-либо специальных средств для установки значений функциональных опций: установка значений функциональных опций производится установкой значений соответствующих констант, редактированием элементов справочников или записей регистров сведений. В конфигурации следует предусмотреть соответствующую функциональность.

2.2. При использовании функциональных опций с параметрами, следует иметь в виду, что если в справочнике или регистре сведений нет записи, соответствующей параметру, то функциональная опция считается выключенной. Если же параметру соответствует больше, чем одна запись, то значения функциональной опции складываются по «ИЛИ».

2.3. Если функциональная опция «привязана» к ресурсу периодического регистра сведений, то система использует срез последних для получения значения опции. Если требуется получать значение опции на какую-либо другую дату, необходимо указать значение для параметра функциональной опции **Период** типа **Дата**, который будет использоваться как дата получения среза. Например, если имеется периодический регистр сведений с измерением **Организация**, то следует использовать следующий синтаксис:

УстановитьПараметрыФункциональныхОпцийФормы(Новый Структура("Организация, Период", <ТребуемаяОрганизация>, <ТребуемаяДата>));

При этом

* значение параметра **Период** необходимо предварительно привести к интервалу периодичности регистра для выполнения требования **2.5**. Например, если периодичность регистра – месяц, то:

НачалоМесяца(<ТребуемаяДата>)

* а сам параметр **Период** не следует создавать в метаданных, так как он предоставляется системой автоматически.

2.4. Также необходимо иметь в виду, что установка значения функциональной опции не вызывает автоматического изменения пользовательского командного интерфейса. Для отработки изменения следует вызвать метод **ОбновитьИнтерфейс**.

2.5. С точки зрения производительности системы следует иметь в виду, что значения функциональных опций кешируются на сервере. Однако большой размер кеша может ухудшить производительность. Поэтому не рекомендуется параметризовать функциональные опции такими данными, которые заведомо могут иметь большое число значений. Например, параметризация функциональной опции контрагентом или товаром не допустима, так как контрагентов или товаров может быть большое количество. Кроме того, зависимость применения функциональности от контрагента сомнительна. На практике функциональность ставится в зависимость от вида, контрагента или иного его признака. Например, если в конфигурации существует перечисление **ВидКонтрагента**, то применение той или иной функциональности следует ставить в зависимость от вида контрагента, а не от самого контрагента.

#### Зависимые функциональные опции

3.1. В некоторых случаях та или иная функциональность должна быть доступна при условии использования или отказа от использования другой функциональности. В подобных случаях сложной зависимости значения функциональной опции от значений других функциональных опций необходимо обеспечить непротиворечивость данных, связанных с функциональными опциями.   
  
Например, функциональность перевода сотрудников из одной организации в другую (т.е. все связанные с этим документы и отчеты) доступна в случае, когда одновременно доступны функциональность "многофирменный учет" и функциональность "кадровый учет".  
  
В таком случае, все объекты метаданных, связанные с переводом сотрудников, не могут и не должны ставиться в зависимость от функциональных опций "многофирменный учет" и "кадровый учет". Для этого необходимо ввести функциональную опцию "перевод сотрудников" и поставить в зависимость от нее все объекты метаданных, для которых это необходимо.  
  
Кроме того, необходимо обеспечить зависимость значения этой функциональной опции от значений "многофирменный учет" и "кадровый учет", например, при записи значений соответствующих констант.  
  
Значения всех трех приведенных в примере функциональных опций рекомендуется показывать администратору системы в соответствующей форме настроек. При этом значение функциональной опции "перевод сотрудников" должно быть недоступно для редактирования.  
  
Редактировать значения таких функциональных опций рекомендуется элементами управления "Поле" вида "Поле флажка" с заголовком, совпадающим с названием соответствующей функциональной опции.  
  
3.2. Значения взаимоисключающих функциональных опций, рекомендуется редактировать в соответствующей форме настройки при помощи элементов управления "Поле переключателя" , "Поле ввода" (со списком выбора) или иной элемент управления, предназначенный для выбора одного значения из многих. При этом, заголовки для переключателей или значения выпадающего списка для "Поле ввода" должны совпадать с названиями функциональных опций.  
  
3.3. В том случае, если та или иная незначительная функциональность сложным образом зависит от значений функциональных опций, но при этом не может быть названа так, чтобы ее название было понятно конечному пользователю, рекомендуется воздержаться от создания очередной функциональной опции. При этом, например, зависимость тех или иных элементов форм должна обеспечиваться при создании формы на сервере за счет анализа значений функциональных опций из кода на встроенном языке.

#### Ограничения на использование параметров функциональных опций

4.1. По соображениям производительности не рекомендуется заводить в конфигурации более 10 параметров функциональных опций. Для того чтобы контролировать их количество в конфигурации, не следует создавать различные параметры функциональных опций одной смысловой нагрузки. Например, вместо двух параметров:

* **ТипВерсионируемогоОбъекта**, связанный с измерением **ТипОбъекта** регистра сведений **НастройкаВерсионированияОбъектов**
* **ТипОбъектаСДополнительнымиОтчетамиИОбработками**, связанный с измерением **ТипОбъекта** регистра сведений **НазначениеДополнительныхОбработок**

рекомендуется создать один параметр функциональных опций **ТипОбъектаКонфигурации**, который связан с измерениями обоих регистров сведений.

4.2. В общем виде, для принятия решения по поводу состава функциональных опций и их параметров рекомендуется придерживаться следующей схемы:

1. Определяется, какая функциональность в нашем прикладном решении может быть опциональной (у нее есть «выключатель»).
2. По каждому выявленному случаю определяется, выключается ли эта функциональность сразу для всей информационной системы или «выключателей» должно быть несколько, например, по одному на каждую организацию или на каждый вид товара.
3. Выписываем список всех параметризуемых функциональных опций, а также список их параметров.
4. При этом в списке параметров функциональных опций не допускаем нескольких параметров одного типа (все функциональные опции, зависящие от организации должны использовать один параметр функциональной опции).
5. Если параметров функциональных опций оказывается неприемлемо много, то составляем их «рейтинг»: суммируем состав всех функциональных опций, которые параметризуются данным параметром и принимаем во внимание важность параметризуемых функциональных опций.
6. Исключаем менее востребованные параметры.
7. Те функциональные опции, которые «лишились» параметров:

* либо делаем непараметрическими (т.е. они включают функциональность во всей информационной базе в целом);
* либо удаляем, если управлять такой функциональностью в целом по информационной базе не имеет смысла.

В результате такого подхода, в конфигурации окажется приемлемое количество параметров функциональных опций.

См. также

* [Влияние изменения значений параметров сеанса и функциональных опций на производительность механизма ограничения доступа к данным](https://its.1c.ua/db/content/v8std/src/600/i8100491.htm?_=1579516850)

### Использование параметров сеанса

#std413

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Параметры сеанса предназначены для хранения значений определенных типов для каждого клиентского сеанса на время работы этого сеанса. Инициализацию параметров сеанса следует выполнять в модуле сеанса (см. ниже раздел 2.1), а их значения рекомендуется использовать в запросах и условиях ограничения доступа к данным для текущего сеанса.  
Примеры параметров сеанса:

* **ТекущийПроект**– тип **СправочникСсылка.Проекты**;
* **ОбменДаннымиВключен** – тип **Булево**;
* **РабочееМестоКлиента**– тип **СправочникСсылка.РабочиеМеста**.

Параметры сеанса доступны из встроенного языка **1С:Предприятия**, например:

Проект = ПараметрыСеанса.ТекущийПроект;

В этом случае, для установки или получения значения параметра сеанса текущий пользователь должен быть наделен соответствующим правом.

Также они могут использоваться в текстах ограничений доступа, например:

ГДЕ Документ.Автор = &ТекущийПользователь

В последнем случае для получения значения параметра сеанса наличия у текущего пользователя соответствующего права не требуется.

1.2.Не рекомендуется использовать параметры сеанса для хранения значений, используемых исключительно в клиентской логике. Поскольку в клиент-серверном варианте **1С:Предприятия** параметры сеанса хранятся на сервере, то любое их считывание или изменение в процессе работы на клиенте потребует дополнительного серверного вызова и увеличит объем передаваемых данных с клиента на сервер и обратно.

В таких случаях следует использовать глобальные переменные модуля управляемого приложения (и обычного приложения – для режима обычного приложения, соответственно).

1.3. Также не рекомендуется использовать параметры сеанса для кеширования вычисленных значений, которые многократно используются в серверной бизнес-логике. В таких случаях следует определять функцию в серверном общем модуле с повторным использованием возвращаемых значений. Исключение составляют случаи, когда время вычисления результата функции модуля с повторным использованием возвращаемых значений соизмеримо с периодом сброса платформенного кеша.

#### Установка параметров сеанса "по требованию"

2.1. Не следует производить инициализацию параметров сеанса при запуске программы, так как:

* не все параметры сеанса запрашиваются из кода конфигурации при запуске программы.
* при работе программы возможно намеренное обнуление значений параметров сеанса из кода на встроенном языке.

Правильным способом установки значений параметров сеанса является установка значений "по требованию" в обработчике **УстановкаПараметровСеанса** модуля сеанса. Т.е. параметры сеанса должны быть инициализированы только в тот момент, когда к ним происходит первое обращение, как к неустановленным.  
  
Пример установки параметров сеанса "по требованию":

Процедура УстановкаПараметровСеанса(ИменаПараметровСеанса)  
   
 Если ИменаПараметровСеанса = Неопределено Тогда  
  // Раздел установки параметров сеанса при начале сеанса (ИменаПараметровСеанса = Неопределено)  
  // Выполняется установка параметров сеанса, которые можно инициализировать  
  // при начале работы системы  
    
 Иначе  
  // Установка параметров сеанса "по требованию"  
    
  // Параметры сеанса, инициализация которых требует обращения к одним и тем же данным  
  // следует инициализировать сразу группой. Для того, чтобы избежать их повторной инициализации,  
  // имена уже установленных параметров сеанса сохраняются в массиве УстановленныеПараметры  
  УстановленныеПараметры = Новый Массив;  
  Для Каждого ИмяПараметра Из ИменаПараметровСеанса Цикл  
   УстановитьЗначениеПараметраСеанса(ИмяПараметра, УстановленныеПараметры);  
  КонецЦикла;  
    
 КонецЕсли;  
   
КонецПроцедуры

// Установить значения параметров сеанса и возвратить имена установленных параметров сеанса  
// в параметре УстановленныеПараметры.  
//  
// Параметры  
//  ИмяПараметра  - Строка - имя параметра сеанса, который требуется установить (проинициализировать).  
//  УстановленныеПараметры  - Массив - массив, в который добавляются имена  
//                                     установленных (проинициализированных) параметров.  
//  
Процедура УстановитьЗначениеПараметраСеанса(Знач ИмяПараметра, УстановленныеПараметры)  
   
 // Если в данном вызове УстановкаПараметровСеанса параметр ИмяПараметра уже  
 // был установлен - возврат.  
 Если УстановленныеПараметры.Найти(ИмяПараметра) <> Неопределено Тогда  
  Возврат;  
 КонецЕсли;  
   
 Если ИмяПараметра = "ТекущийПользователь" Тогда  
   ПараметрыСеанса.ТекущийПользователь = <значение>;  
   ПараметрыСеанса.<другой параметра сеанса> = <значение>;  
   УстановленныеПараметры.Добавить(ИмяПараметра);  
   УстановленныеПараметры.Добавить("<другой параметра сеанса>");  
 КонецЕсли;

КонецПроцедуры

См. также

* [Влияние изменения значений параметров сеанса и функциональных опций на производительность механизма ограничения доступа к данным](https://its.1c.ua/db/content/v8std/src/600/i8100491.htm?_=1579516850)

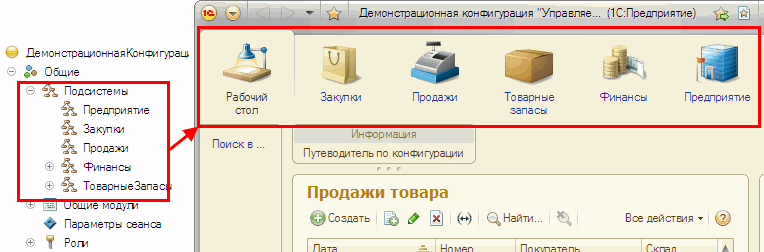
### Использование подсистем

#std543 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, обычное приложение.

1.1. С помощью подсистем решаются две методические задачи:

* Сформировать глобальный командный интерфейс основного окна приложения, которое дает пользователю представление о функциональности приложения в целом.
* Сгруппировать объекты метаданных по функциональному признаку для удобства разработки.

В простейшем случае, получившаяся для обоих задач структура подсистем конфигурации может совпадать.

Например, видимые для пользователей разделы командного интерфейса «Закупки», «Продажи» и пр. могут использоваться одновременно и при разработке: для быстрого отбора объектов в окне метаданных Конфигуратора, при переносе объектов в другие конфигурации, для задания ограничений области поиска при глобальном поиске по конфигурации и т.д.

У таких подсистем должен быть установлен флажок **Включать в командный интерфейс**.

1.2. В общем случае, подсистема, логически объединяющая некоторый набор объектов метаданных, может не совпадать с одним разделом командного интерфейса приложения. Для логического объединения набора объектов метаданных по функциональному признаку рекомендуется заводить в конфигурации отдельную иерархию подсистем, не включенных в командный интерфейс. У таких «функциональных» подсистем флажок **Включать в командный интерфейс** должен быть снят.

Примеры:

* справочник **Номенклатура** логически относится к одной «функциональной» подсистеме «Нормативно-справочная информация», но доступен в командном интерфейсе одновременно в двух разделах – «Нормативно-справочная информация» и «Маркетинг»
* в раздел командного интерфейса «Настройка и администрирование» помещаются команды открытия списков объектов, логически относящихся к тем «функциональным» подсистемам конфигурации, которые предоставляют возможность настройки для администратора системы.

1.3. При этом, общие модули, регламентные задания, константы, подписки на события и прочие объекты, не имеющие визуального представления в командном интерфейсе, рекомендуется включать только в состав «функциональных» подсистем.

См. также

* [Панель разделов](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/400/i8100567.htm?_=1579516850)

### Использование общих реквизитов

#std677

Область применения: управляемое приложение, обычное приложение.

1. Общие реквизиты позволяют добавлять реквизиты сразу для нескольких объектов метаданных (справочников, документов, регистров и т.п.) для решения одной из двух прикладных задач:

* для разделения данных (свойство **Разделение данных** имеет значение **Разделять**);
* для расширения состава реквизитов у нескольких объектов  (свойство **Разделение данных** = **Не использовать**).

2. Общие реквизиты без разделения данных предназначены для добавления некоторого функционала, который не является частью бизнес-логики прикладных объектов (например, решает задачи конфигурации в целом), но при этом требует хранения некоторых данных непосредственно в самих прикладных объектах (а не, например, в связанных регистрах).

При этом общие реквизиты не предназначены для удобства добавления одинаковых реквизитов в прикладные объекты. Например, неправильно переносить в общие реквизиты «обычные» реквизиты документов **Ответственный**, **Комментарий**, **Организация** и т.п. Следует также иметь в виду, что права доступа к общим реквизитам настраиваются отдельно от тех объектов, в которые они добавлены.

3. Порядок следования в дереве метаданных общих реквизитов-разделителей следует подбирать, исходя из требуемого порядка установки [параметров сеанса](https://its.1c.ua/db/content/v8std/src/100/100/i8100413.htm?_=1579516850), которые связаны с ними.

См. также

* [Использование общих реквизитов без разделения данных](http://its.1c.ru/db/metod81#content:4894:1) (статья на ИТС)

### Использование определяемых типов

#std704

Область применения: управляемое приложение, обычное приложение.

1. Определяемые типы предназначены для определения типов данных, которые описывают часто используемые сущности или с высокой степенью вероятности могут изменяться при внедрении прикладного решения. Они позволяют многократно использовать описываемый тип или набор типов без уточнения состава в различных местах конфигурации (в реквизитах, свойствах объектах, форм и т.п.).

См. также статью на ИТС: [*«Объекты конфигурации – Определяемые типы»*](https://its.1c.ua/db/v83doc/bookmark/dev/TI000000236)

2. Определяемые типы рекомендуется использовать в следующих случаях:

2.1. Для определения простого типа и его квалификаторов, имеющего прикладной смысл, который используется в различных реквизитах, ресурсах, реквизитах форм, макетах и т.д. в рамках какой-либо подсистемы или во всем прикладном решении. Это гарантирует одинаковую длину, точность данных во всех местах использования, упрощает доработку в случае изменения требований.  
Например:

* **НомерСчетаФактуры** - Строка, длина 30. Регламентирует формат номера счета-фактуры в различных документах: **ПоступлениеТоваровИУслуг**, **ЗаписьКнигиПокупок**, **ВозвратТоваровОтКлиента** и др.
* **АдресДоставки** - Строка, 500. Текстовое представление адреса доставки в документах **ЗаказПоставщику**, **АдресДоставкиПеревозчика**, в обработке **ПомощникПродаж**, в реквизите **АдресДоставкиПеревозчика** документа **ЗаявкаНаВозвратТоваровОтКлиента** и др.

2.2. Для определения [составного типа](https://its.1c.ua/db/content/v8std/src/100/300/i8100728.htm?_=1579516850), который массово используется в объектах какой-либо подсистемы или во всем прикладном решении. Определяемый тип гарантирует одинаковый состав (тип) данных во всех местах использования, а также упрощает доработку и внедрение подсистем в прикладные конфигурации.

Например, в конфигурацию внедрена подсистема **Взаимодействия**, которая предназначена для ведения переписки по электронной почте, регистрации звонков и встреч. При внедрении этой подсистемы разработчик принял решение о составе объектов метаданных, которые могут выступать в качестве «контактов взаимодействий» - это элементы справочников **ФизическиеЛица**, **Партнеры**, **КонтактныеЛицаПартнеров**, и задал этот состав типов в определяемом типе **КонтактВзаимодействий**, предусмотренном в подсистеме. В свою очередь, определяемый тип массово используется в реквизитах объектов и формах подсистемы (в документах **Встреча**, **ЗапланированноеВзаимодействие** - табличная часть **Участники**, в документе **СообщениеSMS** – табличная часть **Адресаты**, в документе **ТелефонныйЗвонок** - реквизит **АбонентКонтакт**, в общих формах **АдреснаяКнига**, **ВыборКонтакта** – реквизиты **КонтактыПоПредмету**,в параметре макета **ИерархияВзаимодействийКонтакт** журнала документов **Взаимодействия** и т.д.) В противном случае, без использования определяемого типа **КонтактВзаимодействий**пришлось бы снимать объекты подсистемы с поддержки и задавать требуемый состав типов во всех перечисленных местах.

2.3. При разработке внедряемой подсистемы - для переопределения прикладного типа, который будет уточнен при внедрении.  
Например, тип подсистемы **Поставщики** при внедрении может быть заменен на прикладной тип конфигурации **Контрагенты**.

3. Некорректно использовать определяемые типы для задания «синонима» существующему типу, «подмены» сущностей, для локального (не массового) использования в рамках одной подсистемы (конфигурации) без необходимости внедрения в другие конфигурации, только из соображений легкости доработки. Как правило, это говорит об ошибке проектирования или о методологически неверном выборе исходного имени типа.

Например, в конфигурации предусмотрен справочник **Контрагенты**, ссылки на который имеются в нескольких регистрах сведений, реквизитах форм и других объектах конфигурации. При этом справочник не является ни частью встраиваемой подсистемы, ни прикладной сущностью, которая может быть расширена другими типами. Тогда некорректно заводить дополнительный определяемый составный тип, состоящего из единственного типа **Контрагенты**, на «всякий случай», для «механического» упрощения возможного изменения конфигурации в дальнейшем, поскольку это размывает прикладной смысл сущности.

### Правила создания общих модулей

#std469

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Общие модули создаются для реализации процедур и функций, объединенных по некоторому признаку. Как правило, в один общий модуль помещаются процедуры и функции [одной подсистемы конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100543.htm?_=1579516850) (продажи, закупки) или процедуры и функции сходного функционального назначения (работа со строками, общего назначения).

1.2. При разработке общих модулей следует выбирать один из четырех контекстов выполнения кода:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Тип общего модуля** | **Пример наименования** | **Вызов сервера** | **Сервер** | **Внешнее соединение** | **Клиент (обычное приложение)** | **Клиент (управляемое приложение)** |
| 1. | Серверный | ОбщегоНазначения (или ОбщегоНазначенияСервер) |  | + | + | + |  |
| 2. | Серверный для вызова с клиента | ОбщегоНазначенияВызовСервера | + | + |  |  |  |
| 3. | Клиентский | ОбщегоНазначенияКлиент (или ОбщегоНазначенияГлобальный) |  |  |  | + | + |
| 4. | Клиент-серверный | ОбщегоНазначенияКлиентСервер |  | + | + | + | **+** |

2.1. **Серверные общие модули** предназначены для размещения серверных процедур и функций, не доступных для использования из клиентского кода. В них реализуется вся внутренняя серверная бизнес-логика приложения.  
Для корректной работы конфигурации в режимах внешнего соединения, управляемого и обычного приложений, серверные процедуры и функции следует размещать в общих модулях с признаками:

* **Сервер**(флажок **Вызов сервера** сброшен),
* **Клиент (обычное приложение)**,
* **Внешнее соединение**.

В таком случае гарантируется возможность вызова серверных процедур и функций с параметрами мутабельных типов (например, **СправочникОбъект**, **ДокументОбъект** и т.п.). Как правило, это:

* обработчики подписок на события документов, справочников и т.п., которые принимают в качестве параметра мутабельное значение (объект).
* серверные процедуры и функции, в которые в качестве параметра передается объект из модулей справочников, документов и пр., а также из модулей с подписками на события.

Серверные общие модули называются по [общим правилам именования объектов метаданных](https://its.1c.ua/db/content/v8std/src/100/300/i8100474.htm?_=1579516850).  
Например: **РаботаСФайлами**, **ОбщегоНазначения**

В отдельных случаях для предотвращения конфликта имен со свойствами глобального контекста может быть добавлен постфикс **"Сервер"**(англ. **"Server"**).  
Например: **РегламентныеЗаданияСервер**, **ОбменДаннымиСервер,**ScheduledJobsServer.

2.2. **Серверные общие модули для вызова с клиента** содержат серверные процедуры и функции, доступные для использования из клиентского кода. Они составляют [клиентский программный интерфейс сервера приложения](https://its.1c.ua/db/content/v8std/src/500/i8100678.htm?_=1579516850).  
Такие процедуры и функции размещаются в общих модулях с признаком:

* **Сервер** (флажок **Вызов сервера** установлен)

Серверные общие модули для вызова с клиента называются по [общим правилам именования объектов метаданных](https://its.1c.ua/db/content/v8std/src/100/300/i8100474.htm?_=1579516850) и должны именоваться с постфиксом **"ВызовСервера"**(англ. **"ServerCall"**).  
Например: **РаботаСФайламиВызовСервера, CommonServerCall**

Следует иметь в виду, что экспортные процедуры и функции в таких общих модулях не должны содержать параметров мутабельных типов (**СправочникОбъект**, **ДокументОбъект** и т.п.), так как их передача из (или в) клиентского кода невозможна.

См. также: [Ограничение на установку признака «Вызов сервера» у общих модулей](https://its.1c.ua/db/content/v8std/src/500/i8100679.htm?_=1579516850)

2.3. **Клиентские общие модули** содержат клиентскую бизнес-логику (функциональность, определенную только для клиента) и имеют признаки:

* **Клиент (управляемое приложение**)
* **Клиент (обычное приложение)**

Исключение составляют случаи, когда клиентские процедуры и функции должны быть доступны только в режиме управляемого приложения (только в режиме обычного приложения или только в режиме внешнего соединения). В таких случаях, допустима иная комбинация двух этих признаков.

Клиентские общие модули именуются с постфиксом **"Клиент"**(англ. **"Client"**).  
Например: **РаботаСФайламиКлиент**, **ОбщегоНазначенияКлиент, StandardSubsystemsClient**

См. также: [*минимизация кода, выполняемого на клиенте*](https://its.1c.ua/db/content/v8std/src/400/i8100629.htm?_=1579516850)

2.4. Для того чтобы [избежать дублирования кода,](https://its.1c.ua/db/content/v8std/src/300/200/i8100440.htm?_=1579516850) рекомендуется создавать клиент-серверные общие модули с теми процедурами и функциями, содержание которых одинаково на сервере и на клиенте. Такие процедуры и функции размещаются в общих модулях с признаками:

* **Клиент (управляемое приложение)**
* **Сервер** (флажок **Вызов сервера** сброшен)
* **Клиент (обычное приложение)**
* **Внешнее соединение**

Общие модули этого вида именуются с постфиксом **"КлиентСервер"**(англ. **"ClientServer"**).  
Например: **РаботаСФайламиКлиентСервер**, **ОбщегоНазначенияКлиентСервер, UsersClientServer**

В то же время, как только возникает необходимость ветвить код в клиент-серверных общих модулях на серверный и клиентский, то не следует [использовать для этого инструкции препроцессора](https://its.1c.ua/db/content/v8std/src/300/200/i8100439.htm?_=1579516850). Вместо этого, функциональность, различную для клиента и для сервера, рекомендуется реализовывать по общим правилам в модулях соответствующего типа – см. пп. 2.1 и 2.3. Такое явное разделение клиентской и серверной бизнес-логики продиктовано соображениями повышения модульности прикладного решения, упрощения контроля со стороны разработчика над клиент-серверным взаимодействием и снижением риска ошибок из-за принципиальных отличий требований к разработке клиентского и серверного кода (необходимость минимизации кода, выполняемого на клиенте, разной доступностью объектов и типов платформы и др.). При этом нужно иметь в виду неизбежное увеличение числа общих модулей в конфигурации.

Подробнее см.: [Использование директив компиляции и инструкций препроцессора](https://its.1c.ua/db/content/v8std/src/300/200/i8100439.htm?_=1579516850)

Особым случаем смешанных клиент-серверных модулей являются [модули форм](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100630.htm?_=1579516850) и команд, которые специально предназначены для реализации серверной и клиентской бизнес-логики в одном модуле.

3.1. Имена общих модулей рекомендуется строить по общим правилам именования объектов метаданных. Название общего модуля должно совпадать с названием подсистемы или отдельного механизма, процедуры и функции которой он реализует. Рекомендуется избегать в названиях общих модулей таких общих слов как "Процедуры", "Функции", "Обработчики", "Модуль", "Функциональность" и т.п. и применять их только в исключительных случаях, когда они более полно раскрывают назначение модуля.

Для того чтобы различать общие модули одной подсистемы, которые созданы для реализации процедур и функций, выполняемых в разных контекстах, рекомендуется задавать им постфиксы, описанные ранее в пп. 2.1-2.4.

3.2. Дополнительно к общим модулям могут быть добавлены уточняющие постфиксы.

3.2.1. Для глобальных модулей добавляется постфикс **"Глобальный"**(англ. **"Global"**), , в этом случае постфикс **"Клиент"** добавлять не следует.  
Например: **РаботаСФайламиГлобальный, InfobaseUpdateGlobal**

3.2.2. Модули, выполняющиеся в привилегированном режиме, имеющие признак **Привилегированный**, именуются с постфиксом **"ПолныеПрава"**(англ. **"FullAccess"**).  
Например: **РаботаСФайламиПолныеПрава**

3.2.3. Модули, предназначенные для реализации на сервере или на клиенте функций с повторным использованием возвращаемых значений (на время вызова или на время сеанса), именуются с постфиксом **"ПовтИсп"**(англ. **"Cached"**) и **"КлиентПовтИсп"**(англ. **"ClientCached"**) соответственно.  
Например: **РаботаСФайламиКлиентПовтИсп, UsersInternalCached**

3.2.4. Серверные и клиентские модули библиотечных конфигураций (которые предназначены не для самостоятельного использования, а для разработки других конфигураций) с процедурами и функциями, допускающие изменение своей реализации, именуются с постфиксами **"Переопределяемый"**(англ. **"Overridable"**) и **"КлиентПереопределяемый"**(англ. **"ClientOverridable"**).  
Например: **РаботаСФайламиКлиентПереопределяемый, CommonOverridable**

См. также: [*Переопределяемые и поставляемые объекты библиотеки*](https://its.1c.ua/db/content/v8std/src/800/i8100553.htm?_=1579516850)

См. также

* [Тексты модулей](https://its.1c.ua/db/content/v8std/src/300/100/i8100456.htm?_=1579516850)
* [Структура модуля](https://its.1c.ua/db/content/v8std/src/300/100/i8100455.htm?_=1579516850)

### Работа с пользовательскими настройками

#std557

Область применения: управляемое приложение, обычное приложение.

1.1. Для хранения персональных настроек пользователя следует использовать хранилище общих настроек. Например, чтение и запись значения настройки «Задавать вопрос при выходе из программы» для текущего пользователя реализуется на встроенном языке с помощью объекта **ХранилищеОбщихНастроек**:

ЗначениеНастройки = ХранилищеОбщихНастроек.Загрузить("НастройкиПрограммы", "ЗадаватьВопросПриВыходе");  
ХранилищеОбщихНастроек.Сохранить("НастройкиПрограммы", "ЗадаватьВопросПриВыходе", ЗначениеНастройки);

При этом для хранения настроек пользователя не следует использовать какие-либо другие способы, в частности, другие объекты метаданных (регистры, реквизиты и табличные части справочников и др.), внешние файлы и пр.

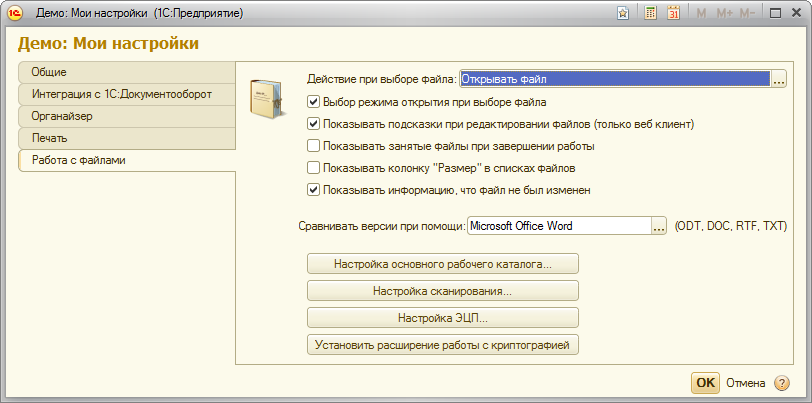
1.2. Для работы с пользовательскими настройками требуется, чтобы для пользователя было доступно право **СохранениеДанныхПользователя**.

См. также: [*Стандартные роли*](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1579516850)

1.3. Для обращения к каждой настройке в хранилище общих настроек необходимо использовать уникальный строковый ключ настройки. Например, значения основной организации и основного склада, с которыми работает текущий пользователь – это две разные настройки, которые хранятся раздельно под ключами **«ОсновнаяОрганизация»** и **«ОсновнойСклад»**.

В то же время, некоторые настройки могут быть объединены в структуру, массив или соответствие, если все обращения к ним выполняются одновременно, как к одной настройке. Например, параметры прокси-сервера для доступа к сети Интернет представляют из себя набор из нескольких значений (адрес прокси-сервер, имя и пароль пользователя), однако хранятся все вместе в виде одной структуры как одна самостоятельная настройка.

2.1. В конфигурации должно быть предусмотрено общее место для редактирования всех пользовательских настроек. Как правило, это общая форма персональных настроек пользователя.

Пример реализации формы персональных настроек «Мои настройки» имеется в демонстрационной конфигурации **Библиотеки стандартных подсистем**.

2.2. В то же время, форма персональных настроек может являться не единственным местом для их редактирования. Для повышения удобства работы пользователя поля с отдельными настройками могут быть размещены непосредственно в тех рабочих местах, к которым эти настройки относятся. Например, флажок «Больше не показывать подсказки при редактировании файла» может быть размещен прямо на форме с самой подсказкой, которая выводится при работе с файлами.

2.3. Форма персональных настроек пользователя, другие формы (рабочие места), а также отдельные элементы форм для работы с персональными настройками должны быть доступны только пользователям с правом **СохранениеДанныхПользователя**.

См. также: [*Стандартные роли*](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1579516850)

3.1. При работе с хранилищем общих настроек следует иметь в виду, что настройки не мигрируют между узлами информационной базы, а специфичны для определенного узла. При необходимости, передача настроек пользователей между узлами может быть реализована дополнительно средствами встроенного языка.

3.2. Все настройки в хранилище общих настроек сохраняются в разрезе пользователей информационной базы, по строковому имени пользователя. Поэтому в случае переименования пользователя прежние настройки теряются. В частности, если впоследствии будет создан пользователь, имя которого совпадает с именем переименованного пользователя, то для него будут использованы ранее сохраненные настройки.

Для того чтобы этого избежать, рекомендуется переносить настройки при переименовании пользователя, и очищать настройки при удалении.

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) в распоряжении разработчика имеются обработчики записи и удаления пользователя информационной базы (см. процедуры **ПриЗаписиПользователяИнформационнойБазы** и **ПослеУдаленияПользователяИнформационнойБазы** в общем модуле **ПользователиПереопределяемый**), в которых возможно выполнить перенос и удаление настроек. Пример использования см. в демонстрационной конфигурации БСП.

### Начальные действия при работе конфигурации

#std556

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В конфигурации должен быть предусмотрен механизм, автоматически определяющий как факт первого запуска конфигурации и выполняющий первоначальное заполнение информационной базы минимально необходимыми данными, так и факт первого запуска нового релиза и выполняющий необходимые изменения в базе (обновление данных ИБ).

При использовании в конфигурации [Библиотеки стандартных подсистем](http://v8.1c.ru/ssl/) такую возможность следует предоставлять средствами подсистемы «Обновление версии ИБ».

В случае если в конфигурации не используется Библиотеки стандартных подсистем, ниже приведены общие требования к начальным действиям, выполняемым при работе конфигурации.

|  |
| --- |
| Методическая рекомендация (полезный совет)  2. Первоначальное заполнение ИБ может быть разделено на обязательное, без чего конфигурация не будет работать, и необязательное, которое облегчает начало эксплуатации продукта, но не является строго обязательным.  3. По результатам обработки информационной базы при первом запуске конфигурации или при первом запуске нового релиза конфигурации рекомендуется выводить администратору системы описание конфигурации или описание изменений в этой версии конфигурации. |

4. Ситуации, когда обработка не проведена в требуемом объеме, должны контролироваться конфигурацией. При этом пользователю должно выводиться предупреждение о возникновении проблемной ситуации. Для вывода подробного протокола о выполненных операциях и возникших ошибках следует использовать журнал регистрации.

5. При наличии в конфигурации возможности работы в распределенной информационной базе (РИБ), следует реализовывать логику обновления данных ИБ в подчиненных узлах таким образом, чтобы она

* выполнялась после загрузки уже обновленных данных из главного узла;
* исключала повторную обработку одних и тех же данных, а тем более – повторное создание новых данных. Другими словами, повторная обработка ИБ должна отрабатывать корректно.

В противном случае:

* если при обновлении подчиненного узла ИБ безусловно создаются новые данные, то они будут многократно созданы в каждом из узлов РИБ и размножены во всех узлах при очередном обмене данными.
* если при обновлении подчиненного узла ИБ безусловно изменяются какие-либо данные, то они будут зарегистрированы к выгрузке обратно в главный узел. Это приведет к дополнительной избыточной нагрузке на канал связи между узлами.

Например, неправильно:

ПрофильОбъект = Справочники.ПрофилиГруппДоступа.СоздатьЭлемент();  
ПрофильОбъект.Наименование = НСтр("ru = 'Бухгалтер'");  
ПрофильОбъект.Предустановленный = Истина;  
ПрофильОбъект.Записать();

правильно:

НаименованиеПрофиля = НСтр("ru = 'Бухгалтер'");  
Запрос = Новый Запрос(  
 "ВЫБРАТЬ  
 | ИСТИНА  
 |ИЗ  
 | Справочник.ПрофилиГруппДоступа КАК ПрофилиГруппДоступа  
 |ГДЕ  
 | ПрофилиГруппДоступа.Наименование = &Наименование И  
 | ПрофилиГруппДоступа.Предустановленный = ИСТИНА");  
Запрос.УстановитьПараметр("Наименование", НаименованиеПрофиля);  
// Если элемента нет, только тогда нужно создать новый.  
Если Запрос.Выполнить().Пустой() Тогда  
  ПрофильОбъект = Справочники.ПрофилиГруппДоступа.СоздатьЭлемент();  
  ПрофильОбъект.Наименование = НаименованиеПрофиля;  
  ПрофильОбъект.Предустановленный = Истина;  
  ПрофильОбъект.Записать();  
КонецЕсли;

### Поддержка толстого клиента, управляемое приложение, клиент-сервер

#std680

Область применения: управляемое приложение, обычное приложение.

1. В управляемом режиме из-за ряда ограничений тонкого клиента может возникнуть необходимость поддержки запуска толстого клиента (в режиме управляемого приложения). Подробнее см. [Функциональность обычного приложения, отсутствующая в управляемом приложении](http://its.1c.ru/db/metod81#content:3266:1).

2. При этом разработка конфигураций, рассчитанных на режим управляемого приложения, как правило, ведется исходя из того, что в клиент-серверной архитектуре код следующих модулей компилируется и выполняется только на сервере

* модуль менеджера;
* модуль объекта;
* модуль сеанса.

В частности, в указанных модулях может встречаться обращение к общим модулям, доступным только на сервере.

Однако в толстом клиенте, в режиме управляемого приложения, клиент-сервер, возможны ситуации, когда указанные модули могут начать компилироваться и выполняться на стороне клиента, в частности:

* если объект (справочник, документ и т.п.) явно создается и вызывается в клиентском коде;
* когда платформа 1С:Предприятие неявно обращается к модулям менеджеров и модулю сеанса для вызова их обработчиков событий на клиенте.

Компиляция и выполнение таких модулей на клиенте могут приводить к ошибкам. По этой причине режим проверки конфигурации для режима толстый клиент, управляемое приложение, может находить ошибки в указанных модулях.

Для того чтобы избежать незапланированной компиляции и исполнения указанных модулей на клиенте, а также чтобы избежать лишних сообщений режима проверки конфигурации, следует:

* полностью исключить из клиентского контекста код модулей объектов (наборов записей и т.п.), заключив его в инструкцию препроцессора и дополнив вызовом исключения, которое предотвращает несанкционированную попытку использования объекта на клиенте:

#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда  
  …  
#Иначе  
  ВызватьИсключение НСтр("ru = 'Недопустимый вызов объекта на клиенте.'");  
#КонецЕсли

* полностью исключить из клиентского контекста код модуля сеанса, заключив его в инструкцию препроцессора (так как параметры сеанса требуются для работы серверного, а не клиентского кода конфигурации):

#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда  
  …  
#КонецЕсли

* полностью исключить из клиентского контекста код модулей менеджеров всех видов объектов метаданных, заключив его в инструкцию препроцессора:

#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда  
…  
#КонецЕсли

В последнем случае также будет действовать следующее ограничение: если представление объектов формируется [обработчиками событий модуля менеджера **ОбработкаПолученияПредставления** и **ОбработкаПолученияПолейПредставления**](https://its.1c.ua/db/content/v8std/src/100/400/i8100746.htm?_=1579516850), то в толстом клиенте, в режиме управляемого приложения, клиент-сервер, представление будет формироваться по умолчанию, без вызова этих обработчиков, и тем самым будет отличаться от остальных режимов работы. (При этом оставшиеся два обработчика модуля менеджера **ОбработкаПолученияДанныхВыбора** и **ОбработкаПолученияФормы** вызываются всегда только на сервере, поэтому указанное ограничение на них не распространяется.)

|  |
| --- |
| Методическая рекомендация (полезный совет)  3. В тех случаях, когда требуется снять указанное выше ограничение, необходимо дополнительно обеспечить работу на клиенте следующих фрагментов серверного кода:   * обработчиков событий модулей менеджеров **ОбработкаПолученияПредставления** и **ОбработкаПолученияПолейПредставления** * а также код подписок на эти события модулей менеджеров.   Для этого код перечисленных обработчиков событий следует вынести за инструкции препроцессора, указанные в п.2, а обработчики подписок разместить в клиент-серверных модулях.  При необходимости вызова серверных процедур (и функций) из клиентского кода следует размещать вызываемые процедуры в [серверных общих модулях с признаком **Вызов сервера**](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1579516850). При этом нужно убедиться, что в параметры процедур (и в возвращаемые значения функций) не передаются значения мутабельных типов (**СправочникОбъект**, **ДокументОбъект** и пр.)  Важно: не следует для этих целей всем общим модулям с признаком **Сервер** принудительно устанавливать флажок **Вызов сервера**. Подробнее см. [Ограничение на установку признака «Вызов сервера» у общих модулей](https://its.1c.ua/db/content/v8std/src/500/i8100679.htm?_=1579516850).  Например, обработчик события **ОбработкаПолученияПредставления** вызывает общий модуль, который не доступен на клиенте:  Процедура ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка)     Взаимодействия.ОбработкаПолученияПредставления(Данные, Представление);   СтандартнаяОбработка = Ложь;   КонецПроцедуры  правильно выполнить переход на сервер (и при этом не передавать на клиент значения мутабельных типов):  Процедура ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка)    ВзаимодействияВызовСервера.ОбработкаПолученияПредставления(Данные, Представление);   СтандартнаяОбработка = Ложь;  КонецПроцедуры |

4. Для расстановки фрагментов кода с инструкциями препроцессора можно воспользоваться [приложенной обработкой](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%90%D0%B2%D1%82%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%9A%D0%BE%D0%B4%D0%B0%D0%98%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F/%D0%90%D0%B2%D1%82%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%9A%D0%BE%D0%B4%D0%B0%D0%98%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F.zip).

### Технология разветвленной разработки конфигураций

#std709 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Цели внедрения технологии:

* Повышение качества разрабатываемой конфигурации
* Повышение культуры разработки и тестирования
* Обеспечение непрерывного развития конфигураций в условиях жестких сроков разработки

#### 1. Определения

**Плановая версия конфигурации** – версия, содержащая существенное развитие функционала, срок выпуска которой назначается заранее.  
  
**Исправительная версия** – версия, которая выпускается при необходимости срочной публикации исправлений критичных ошибок. В исключительных случаях исправительная версия может содержать какой-то новый функционал (например, доработки, связанные с поддержкой изменения законодательства). Срок выпуска определяется при анализе количества и критичности обнаруженных ошибок плановой версии.  
  
**Технический проект** – задание на доработку конфигурации. Каждый технический проект имеет четко сформулированную цель и конечный список изменений, которые нужно выполнить, чтобы достигнуть этой цели.

Для организации работ по разработке и сопровождению конфигураций (в т.ч. ведению информации о технических проектах и списка ошибок) рекомендутся использовать **Систему проектирования прикладных решений (СППР)**.

#### 2. Разработка исправительных версий

2.1. Для выпуска каждой исправительной версии создается новое хранилище на основе конфигурации последней выпущенной версии.  
  
Важно – нужно создавать новое хранилище, а не копировать основное!

2.2. В исправительной версии не должно быть объемных доработок конфигурации, в противном случае нужно пересматривать сроки выпуска плановой версии.

2.3. Все закладки в хранилище исправительной версии должны содержать комментарий.  
  
Требования к содержанию комментариев аналогичны требованиям к закладкам в хранилище плановой версии (см. п.3.4).

2.4. Все изменения, которые выполняются в исправительном релизе, должны синхронно повторяться в основном хранилище. Если в исправительном релизе добавляются новые объекты (реквизиты объектов), то переносить изменения нужно исключительно с помощью сравнения/объединения конфигураций, для того чтобы не различались внутренние идентификаторы объектов конфигурации.

2.5. При сборке исправительной версии рекомендуется устанавливать метку с информацией о номере сборки на закладке той версии хранилища, конфигурация которой идет в сборку. Обычно это последняя на момент сборки закладка.

#### 3. Разработка плановой версии

3.1. Разработка плановых версий ведется в основном хранилище конфигурации.

3.2. Закладки в основное хранилище должны осуществляться таким образом, чтобы каждая закладка переводила конфигурацию хранилища из одного рабочего (готового к выпуску) состояния в другое.  
**Не допускается закладка не полностью отлаженного функционала!** Основное хранилище всегда должно находиться в «неразваленном» состоянии, чтобы в любой момент можно быть начать сборку плановой версии.

3.3. В основном хранилище разрешается выполнять следующие работы:

* исправление ошибок, не требующих перепроектирования, объемного кодирования и тестирования. Если ошибка требует больших переработок и/или пересмотра проектных решений, то исправление такой ошибки должно вестись в рамках технического проекта. Порядок работы с основным хранилищем должен быть таким же, как и по другим техническим проектам;
* встраивание новых версий библиотек;
* встраивание полностью отлаженных, прошедших отладочное тестирование проектов;
* в исключительных случаях в основном хранилище может вестись разработка некоторых проектов, (например, проектов по массовому рефакторингу).

Рекомендуется использовать реализованные в **СППР** возможности автоматической генерации текстов комментариев для закладок, связанных с исправлением ошибок и встраиванием технических проектов.

3.4. Все закладки в основное хранилище должны содержать комментарий.  
  
Содержание комментария зависит от характера выполненных работ:

* при исправлении ошибки обязательно должен быть указан номер и краткое наименование ошибки в системе баг-трекинга;
* при встраивании новой версии библиотеки должно быть указано название библиотеки и точный номер версии библиотеки;
* при встраивании технических проектов – номер проекта в системе ведения проектной документации, а также краткое наименование;
* при выполнении работ по техническому проекту в основном хранилище комментарий, помимо номера и краткого наименования проекта, должен содержать краткое описание сделанных в этой закладке изменений.

3.5. Все изменения по техническому проекту должны переноситься в основное хранилище за одну закладку. Если необходимо переносить изменения несколько раз, то нужно открывать несколько проектов.

3.6. После переноса изменений в основном хранилище можно исправлять ошибки, наведенные техническим проектом. Для пересмотра проектных решений нужно открывать новый проект.

3.7. При сборке плановой версии рекомендуется устанавливать метку с информацией о номере сборки на закладке той версии хранилища, конфигурация которой идет в сборку. Обычно это последняя на момент сборки закладка.

#### 4. Разработка технических проектов

4.1. Разработка каждого технического проекта ведется в отдельном хранилище.

При использовании **СППР** хранилище технического проекта может быть созданно автоматически. Если **СППР** не используется, хранилище технического проекта нужно будет создавать вручную, в соотвествии с порядком, описанном в Приложении 1.

4.2. При постановке хранилища технического проекта на поддержку от основного хранилища, платформа для всех объектов устанавливает правило «Объект поставщика, не редактируется». Для работы над техническим проектом нужно изменить это правило на «Объект поставщика редактируется с сохранением поддержки».  
  
Правило «Объект поставщика редактируется с сохранением поддержки» нужно устанавливать только для тех объектов, которые изменяются при выполнении технического проекта. Правило нужно менять как можно более точечно – например, если изменения в проекте будут затрагивать только форму, то нужно изменить правило только для этой формы, а для объекта, которому эта форма принадлежит, нужно оставить правило «Объект поставщика, не редактируется».  
  
Для изменения правил поддержки нужно захватить только корень конфигурации, захватывать сами объекты не нужно.  
  
Выполнение этих рекомендаций позволит упростить процесс переноса изменений между основным хранилищем и хранилищем тех. проекта.

4.3. Ответственный за технический проект может периодически обновлять конфигурацию хранилища проекта. Периодичность обновления разработчик определяет самостоятельно.  
  
На частоту обновления могут влиять следующие факторы:

* затрагивает ли технический проект объекты других ответственных;
* проводится ли в данное время рефакторинг общих механизмов;
* ведется ли сейчас в основном хранилище массовое исправление ошибок.

Порядок обновления хранилища технического проекта описан в приложении 2.

4.4. После окончания разработки ответственный согласует сроки завершения отладочного тестирования и сроки внесения технического проекта в основное хранилище. Проекты, затрагивающие большое количество объектов рекомендуется вноситься в основное хранилище ближе к сроку окончания разработки, чтобы уменьшить влияние на другие проекты.  
  
Ответственные за другие технические проекты могут попросить перенести сроки внесения в основное хранилище.

В **СППР** согласовывать сроки встраивания технических проектов можно, используя функциональность контрольных точек по техническому проекту.

4.5. Внесение проекта в основное хранилище должно осуществляться после завершения отладочного тестирования. Рекомендуется по окончании исправления ошибок, выявленных отладочным тестированием технического проекта, сформировать файл сравнения конфигурации проекта и конфигурации основного хранилища.

4.6. Внесение наработок технического проекта в основное хранилище не должно проводить к длительному захвату объектов основного хранилища. Это достигается тем, что сначала хранилище технического проекта обновляется до состояния основного хранилища (по методике, описанной в приложении 2). Если изменений много, то такое обновление может занять достаточно много времени (до нескольких дней) – за это время конфигурация основного хранилища может измениться. Поэтому процесс обновления может быть итеративным – на каждой итерации обновления отличия в конфигурациях будут становиться все ближе к величине изменений, внесенных техническим проектом.  
  
После каждой итерации обновления целесообразно проводить быструю проверку работоспособности функционала, разрабатываемого в рамках проекта.  
  
Начинать перенос изменений в основное хранилище (захватывать объекты в основном хранилище) следует только тогда, когда конфигурация технического проекта будет отличаться от конфигурации основного хранилища практически только на изменения, вносимые проектом.

4.7. Ответственный за технический проект должен внимательно относиться к внесению изменений в основное хранилище. Нужно помнить, что основное хранилище должно в любой момент времени находиться в состоянии готовности к выпуску плановой версии.  
  
После внесения изменений в основное хранилище разработчики технического проекта совместно с тестировщиками проводят быструю проверку того, что изменения перенесены корректно и не повлияли на работоспособность смежного функционала. Объем проверок и порядок их проведения определяет ответственный за проект.

4.8. После проверки переноса изменений и до закладки изменений в основное хранилище, ответственный обязательно должен запустить проверку конфигурации. Проверку нужно проводить с максимальными настройками.  
Закладка изменений в основное хранилище допускается только после того, как будут исправлены все ошибки, выявленные проверкой конфигурацией, которые были привнесены проектом.

4.9. После переноса изменений в основное хранилище ответственный за технический проект удаляет хранилище проекта

#### 5. Нумерация сборок

Изменение номеров версий регламентируется стандартом [Нумерация редакций и версий](https://its.1c.ua/db/content/v8std/src/100/200/i8100483.htm?_=1579516850)  
Здесь будут уточнены правила изменения номера сборки (четвертое число в номере версии)

5.1. Номер сборки следует увеличивать как в основном хранилище, так и в хранилище исправительного релиза в двух случаях:

* непосредственно перед сборкой релиза. Это необходимо, чтобы полный номер собранного релиза гарантированно отличался от полного номера предыдущего релиза;
* при закладке в хранилище обработчика обновления информационной базы. Это необходимо, чтобы после обновления из хранилища у всех участников разработки добавленный обработчик обновления запускался автоматически (только для конфигураций, основанных на **Библиотеке Стандартных Подсистем**).

5.2.1. При добавлении в хранилище обработчиков обновления информационной базы рекомендуется в рамках этой же закладки повышать номер сборки. Существует два возможных сценария:

* Обработчик обновления добавляется при разработке технического проекта в хранилище технического проекта. В этом случае при переносе изменений в основное хранилище следует увеличить номер сборки основного хранилища.
* Обработчик обновления добавляется в рамках исправления ошибки. Если ошибка исправляется только в одном хранилище (основном или исправительном), то номер сборки повышается только в нем, если в двух – значит нужно увеличить номер в обоих хранилищах.

5.2.2. Обработчик и изменение номера сборки должны помещаться в хранилище в рамках одной закладки. При этом обработчик обновления должен быть «привязан» к тому номеру сборки, который вместе с ним помещается в хранилище.

5.2.3. Если в рамках одной конфигурации обработчики обновления разбиты по технологическим подсистемам (например, в конфигурации **1С:ERP**обработчики разбиты на подсистемы УправлениеПредприятием и УправлениеТорговлей), то нужно повышать номер сборки как подсистемы, к которой относится обработчик, так и конфигурации.

5.3. Номер сборки необходимо изменять:

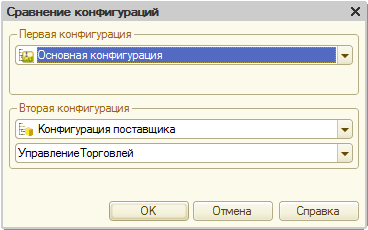
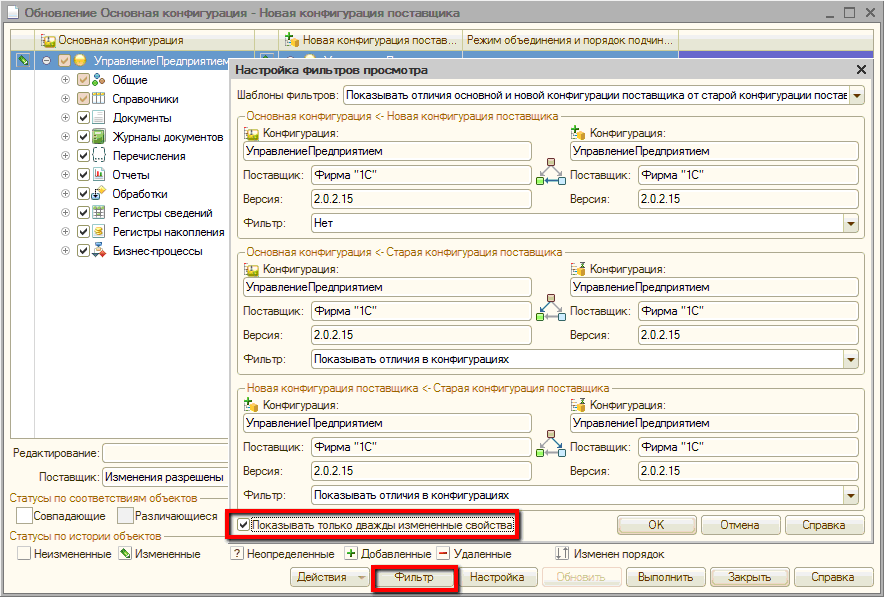
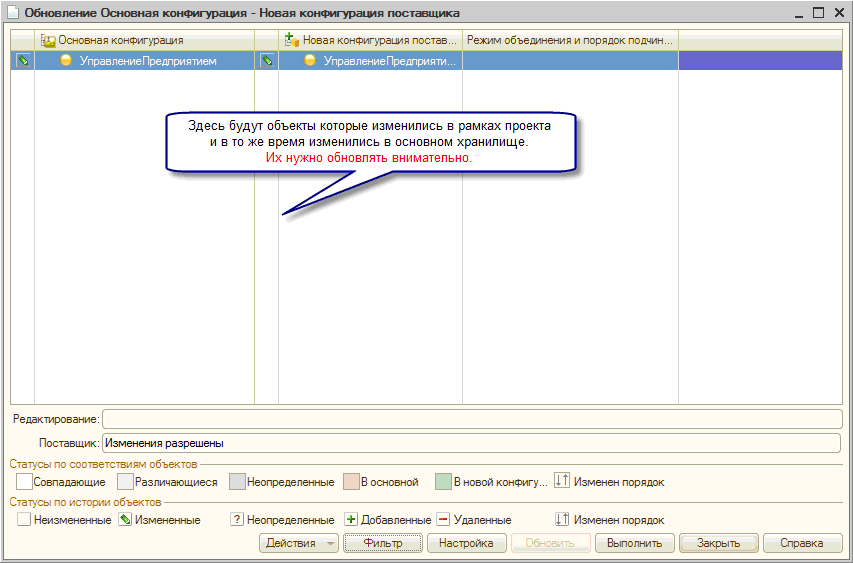
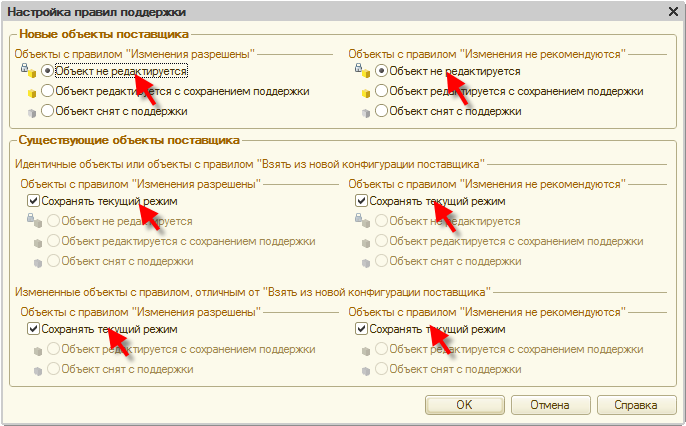
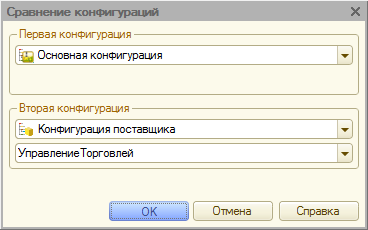
1. В свойствах конфигурации.
2. В процедуре ОбновлениеИнформационнойБазы<ИмяБиблиотеки>.ПриДобавленииПодсистемы (только для конфигураций, основанных на **Библиотеке Стандартных Подсистем**).

#### Приложение 1. Порядок создания хранилища технического проекта

1. Обновить из хранилища конфигурацию информационной базы, подключенную к основному хранилищу
2. Создать файл поставки конфигурации основного хранилища (\*.cf)
3. В информационную базу, которая будет использоваться для работы над техническим проектом, **загрузить** конфигурацию из файла поставки. После загрузки конфигурации из файла поставки конфигурация будет находиться на поддержке без возможности изменения.
4. Создать хранилище конфигурации в соответствующей общей папке (при создании хранилища платформа включит в конфигурации возможность изменения)
5. Добавить пользователя ТолькоПросмотр (без пароля, без права захвата объектов). Этого пользователя не нужно использовать для подключения базы к хранилищу – только для обновления из хранилища (получения конфигурации хранилища)
6. Добавить в хранилище пользователей, перечисленных в проекте (логин – фамилия сотрудника, без пароля, с правом захвата объектов). Не нужно использовать для работы участников проекта логин пользователя ТолькоПросмотр.

#### Приложение 2. Порядок обновления хранилища технического проекта до состояния основного хранилища

Перед выполнением переноса изменений из хранилища технического проекта (далее ХТП) в основное хранилище (далее ОХ) выполняется обновление ХТП до состояния ОХ.  
  
Для того чтобы обновить ХТП до состояния ОХ необходимо выполнить следующее:

1. Обновить информационную базу, подключенную к ОХ.
2. Создать файл поставки конфигурации ОХ.
3. Захватить все объекты в ХТП.
4. Запустить сравнение основной конфигурации и конфигурации поставщика (Конфигурация – Сравнить конфигурации). Результаты сравнения сохранить в файл – это изменения, внесенные в конфигурацию при работе над техническим проектом. В меню «Действия» выбрать пункт «Отчет о сравнении конфигураций». Для дальнейшего использования лучше вывести и сохранить отчет о сравнении и в текстовом формате и формате табличного документа.  
     
   
5. Обновить конфигурацию (Конфигурация – Поддержка – Обновить конфигурацию – Выбор файла обновления – указать файл поставки конфигурации созданный на шаге 2).  
     
   В появившемся окне сравнения и объединения конфигураций нажать кнопку "Фильтр" и установить флажок Показывать только дважды измененные свойства".  
     
     
     
   На эти объекты нужно обратить внимание при объединении, остальные изменения можно объединять без проверки.  
     
   
6. В диалоге, который появляется при нажатии на кнопку «Выполнить» окна сравнения и объединения конфигураций, для новых объектов поставщика нужно установить правило «Объект не редактируется» - как для объектов с правилом «Изменения разрешены» так и для объектов с правилом «Изменения не рекомендуются», для всех остальных установить флаг «Сохранять текущий режим» (по умолчанию он установлен).  
     
   
7. После завершения объединения нужно исправить объекты, затрагиваемые техническим проектом, изменения в которых затерлись при обновлении. По сути это означает, что нужно выполнить повторное внесение доработок, реализованных в рамках технического проекта в объекты конфигурации.
8. Запустить сравнение обновленной основной конфигурации технического проекта и обновленной конфигурации поставщика (Конфигурация – Сравнить конфигурации)  
     
   
9. Результаты сравнения сохранить в файл, имя файла должно отличаться от имени файла созданного на шаге 6. В меню «Действия» выбрать пункт «Отчет о сравнении конфигураций». Для дальнейшего использования лучше вывести и сохранить отчет о сравнении в текстовом формате.
10. Сравнить файлы, созданные на шаге 4 и шаге 9. При правильном обновлении, сравнение файлов не должно показать отличий.

### Особенности разработки конфигураций для ОС Linux и macOS

#std723

Область применения: управляемое приложение, обычное приложение.

1. В большинстве случаев, в конфигурации не требуется предпринимать каких-либо специальных мер для обеспечения работы конфигурации (клиентское приложение и сервер) на ОС Linux и macOS. В этой статье перечислены отдельные рекомендации для специфических случаев, описанных в [приложении 7 документации по платформе 1С:Предприятие](http://its.1c.ru/db/v83doc#content:97:1).

2. Для реализации всех ключевых функций прикладного решения следует использовать возможности платформы 1С:Предприятие по унификации работы на различных операционных системах.

2.1. Вместо Windows-технологии COM (объект **COMОбъект**) следует использовать специализированные кроссплатформенные механизмы платформы:

* Для администрирования кластера серверов 1С:Предприятия, вместо работы с объектной моделью агента сервера через COM-объект **v83.ComConnector**, следует использовать сервер администрирования (ras) и утилиту администрирования (rac). При работе в macOS утилиты rac и ras недоступны.
* Для получения путей к рабочим каталогам, вместо COM-объектов ОС Windows, следует использовать методы глобального контекста **РабочийКаталогДанныхПользователя**, **КаталогДокументов**, **КаталогВременныхФайлов**.

В остальных случаях следует рассмотреть другие альтернативы технологии COM, работающие в ОС Linux и macOS, например, технологию создания внешних компонент Native API.

2.2. Внешние компоненты (клиентские и серверные), поставляемые в составе конфигурации, следует разрабатывать с использованием технологии Native API. Это позволяет создавать кроссплатформенные внешние компоненты для различных операционных систем, а также для веб-клиента, работающего в веб-браузерах, которые поддерживаются платформой 1С:Предприятие. Подробнее о разработке внешних компонент см. [документацию по платформе](http://its.1c.ru/db/v83doc#content:90:1).

2.3. Для механизмов, использующих объект **Почта**, следует рассмотреть альтернативные варианты:

* По переводу на объект **ИнтернетПочта**;
* По разработке внешних компонент для ОС Linux и macOS, которые поддерживают работу с установленными почтовыми клиентами в ОС Linux и macOS.

2.4. Если в составе конфигурации поставляются картинки в форматах WMF и EMF (метафайлы Windows), их следует заменить на растровые, например PNG или JPG.

2.5. Также следует использовать возможности платформы 1С:Предприятие по унификации работы с файловой системой.

2.5.1. В ОС Linux имена файлов регистро-зависимые, поэтому во всех местах кода, который работает с конкретным файлом, его имя (путь) должен указываться в одном регистре.

2.5.2. Не следует указывать разделить пути файла и маску всех файлов вручную (например, «/», «\*.\*»), для этого необходимо использовать функции **ПолучитьРазделительПути** и **ПолучитьМаскуВсеФайлы**.

При использовании в конфигурации **Библиотеки стандартных подсистем** для работы с именами файлов также рекомендуется использовать функции общих модулей **ОбщегоНазначения**и **ОбщегоНазначенияКлиент**.

3. Для отдельных второстепенных (сервисных) функций прикладного решения допустимо отключать их работу в ОС Linux и macOS. Например, для прикладного решения в области торгового учета второстепенными могут считаться возможности по синхронизации данных через прямое подключение с другими программами, по импорту почты из сторонних почтовых клиентов и т.п.

Для этого следует скрывать команды таких механизмов из командного интерфейса программы при работе в ОС Linux и macOS, либо (если технически скрыть невозможно) выводить сообщение вида  
«<Операция> доступна только при работе в ОС Windows».

Например:

&НаКлиенте  
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)  
    Информация = Новый СистемнаяИнформация;  
    Если Информация.ТипПлатформы <> ТипПлатформы.Windows\_x86 И Информация.ТипПлатформы <> ТипПлатформы.Windows\_x86\_64 Тогда  
        ПоказатьПредупреждение(, НСтр("ru = 'Печать в Microsoft Word доступна только при работе в ОС Windows.'"));  
        Возврат;  
    КонецЕсли;  
     
    <...>  
КонецПроцедуры

При использовании в конфигурации **Библиотеки стандартных подсистем** рекомендуется использовать функции **ЭтоLinuxКлиент**, **ЭтоMacOSКлиент** и **ЭтоWindowsКлиент** из общих модулей **ОбщегоНазначения** и **ОбщегоНазначенияКлиент.**

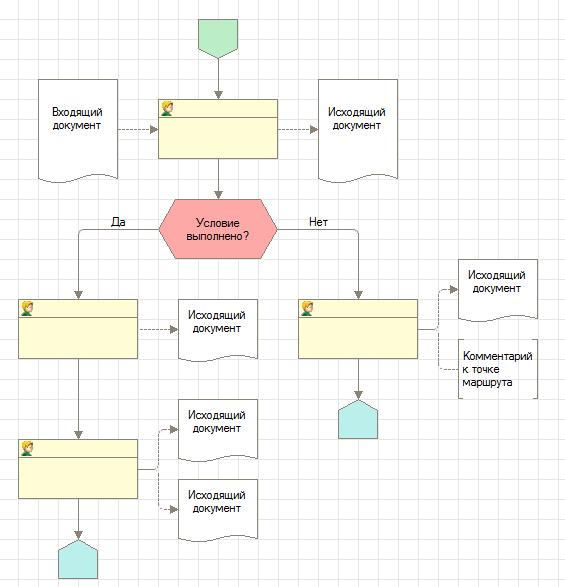
См. также

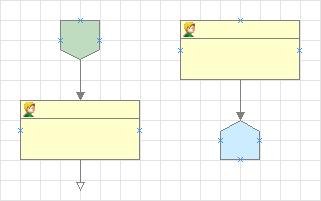
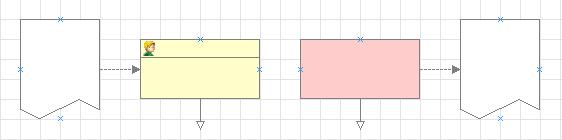
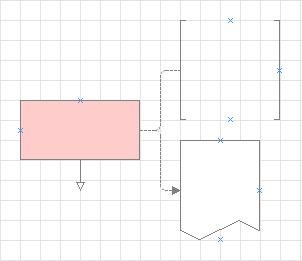
* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1579516850)
* [Глава "35. Особенности разработки кроссплатформенных прикладных решений"](http://its.1c.ru/db/v837doc#bookmark:dev:TI000001208) документации по платформе 1С:Предприятие

### Оформление карты маршрута бизнес-процесса

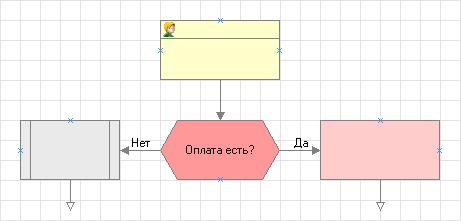
#std480

Область применения: управляемое приложение, обычное приложение.

Пример оформления бизнес-процесса:  


**1. Общие рекомендации**  
  
1.1 Бизнес-процессы рисуются **сверху - вниз**. Другая ориентация (например, слева направо) допустима, если позволяет изобразить бизнес-процесс наилучшим образом.  
  
1.2 Используется стандартная **сетка** – шаг 20 точек, нарисованная линиями.  
  
1.3 Не используются жирные и разноцветные **шрифты**.  
  
1.4 Количество **изгибов** соединительных линий должно быть минимальным. Если можно соединить две точки, изогнув соединительную линию один раз, то нужно стараться сделать именно так, подгоняя размеры и положение точек.  
  
1.5 При размещении декораций и точек маршрута рекомендуется соблюдать отступ в одну клетку от левого и верхнего края карты маршрута.  
  
1.6 **Ширину** точек маршрута нужно делать равной четному количеству клеток. Это необходимо для того, чтобы соединительные линии сходились без дополнительных изгибов. **Размеры** точек и декораций нужно стремиться приближать к пропорциям спичечного коробка (например, 3х6 клеток). Следует избегать вытянутых по вертикали или горизонтали точек – они мешают восприятию схем.  
  
1.7 **Выравнивание текста** в точках маршрута и декорациях следует делать по центру (по умолчанию), за исключением комментариев – их лучше делать выровненными влево, как обычный текст.  
  
1.8 Рекомендуется размещать декорации и точки маршрута так, чтобы они не пересекали границы печатного листа (меню Графическая схема - Режим просмотра страниц).  
  
**2. Рекомендации по оформлению отдельных элементов**  
  
2.1 Точки **Старт** и **Завершение** устанавливаются размером 2х2 клетки, без наименования. Количество точек старта и завершения неограниченно и определяется логикой бизнес-процесса. Отступ от точки старта или завершения до другой точки бизнес-процесса рекомендуется устанавливать в две клетки:  
  
  
2.2 Точки "**Действие**", "**Автоматическая обработка**", "**Вложенный бизнес-процесс**" должны иметь размер сообразно размещаемому в них тексту. Текст должен носить характер короткой директивы и отвечать на вопрос "Что нужно сделать?" (например "Выписать счет", а не "Выписка счета"). Слева от этих точек нужно располагать декорации, описывающие исходные или входящие данные, требуемые для выполнения действий, предусмотренных данной точкой. Справа от точки нужно располагать декорации, описывающие результирующие или исходящие данные, которые получаются в результате выполнения действий, предусмотренных данной точкой.  
  
2.3 Изображения **входящих документов**, служащих основой для выполнения действий, предусмотренных в точках маршрута нужно располагать слева от точки на расстоянии двух клеток и соединять пунктирной линией со стрелкой. **Исходящие документы**, являющие результатом выполнения действий, предусмотренных точкой маршрута, нужно располагать справа от точки на расстоянии двух клеток и соединять пунктирной линией со стрелкой. Документы оформляются фигурой вида «**Документ**»:  
  
  
Пропорции декорации, изображающей документ должны по возможности приближаться к формату А4 (например, 4х5 или 3х4 клетки стандартной сетки).  
  
**2.4 Комментарии** к точкам маршрута нужно изображать на расстоянии двух клеток слева от точки маршрута, если они относятся к входящему документу, или справа от точки маршрута, если к исходящему документу. Комментарий оформляется в виде фигуры "Скобки вертикальные" или "Скобки горизонтальные" и соединяется с левой или правой гранью точки маршрута пунктирной линией без стрелки:  


**2.5** **Точки условного перехода** включают в себя короткий текст вопроса, заканчивающийся знаком вопроса. Вопрос нужно формулировать так, чтобы на него можно было ответить только Да или Нет (например, Счет есть?, Отгрузка разрешена?, ОК?). Точки условия нужно стремиться сделать как можно более компактными, но не в ущерб наглядности. Это связано с тем, что, точки условия являются точками перехода, а не точками действия, и поэтому для участников бизнес-процесса второстепенны. Соединительные линии точек условия (как входящие, так и исходящие) должны иметь суммарную длину не менее двух клеток для того, чтобы надписи Да и Нет читались нормально, например:



### Ограничения на переименование объектов метаданных

#std706

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Запрещается переименовывать общий модуль и создавать новый с тем же именем. Следует создавать новый общий модуль с новым именем, а код исходного модуля переносить в новый модуль.

В противном случае, при сравнении и объединении с новой конфигурацией поставщика, например, в подписках на события, которые ссылаются на общий модуль, будет установлена связь по имени с новым общим модулем (а не переименованным), где нет требуемых процедур.

2. В некоторых случаях, при изменении структуры метаданных возникает необходимость удалить объект метаданных и создать новый с тем же именем. Например, при «сужении» состава измерений в регистре. В таких случаях следует переименовать устаревший объект метаданных, добавив ему к имени префикс Удалить.

В противном случае, если удалить объект метаданных и создать новый с тем же именем, может возникнуть ошибка при обновлении конфигурации на поддержке. Если в конфигурации на поддержке в устаревший объект вносились изменения с сохранением поддержки (например, в целях переопределения поведения объекта или исправления ошибок), то он не будет удален при объединении конфигураций, и попытка обновления конфигурации базы данных завершится неудачно из-за наличия двух объектов метаданных (старого и нового) с одинаковым именем.

См. также [Удаление устаревших объектов метаданных из конфигурации](https://its.1c.ua/db/content/v8std/src/100/300/i8100534.htm?_=1579516850)

|  |
| --- |
| Методическая рекомендация (полезный совет)  3. Если в конфигурации возникает необходимость хранить ссылки на объекты метаданных, то рекомендуется придерживаться следующих правил.  3.1. Если в конфигурации не используется **Библиотека стандартных подсистем** (БСП), то рекомендуется заводить строковые реквизиты (**Строка**, 255) для хранения полных имен объектов метаданных. Например, в справочнике **ШаблоныСообщений** – реквизит **ПолноеИмяТипаПараметраВводаНаОсновании**, в регистре сведений **НастройкиПечатиОбъектов** - измерение **ТипОбъекта** и т.п.  При этом если в конфигурации переименовываются (или удаляются) какие-либо объекты метаданных, то при обновлении информационной базы требуется также предусмотреть соответствующую замену (удаление) всех имен, хранимых в базе. В противном случае, ссылки на имена объектов метаданных станут рассогласованными, что приведет к различным ошибкам в тех подсистемах конфигурации, которые опираются на имена объектов метаданных.  3.2. При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) следует использовать ссылки на справочник **ИдентификаторыОбъектовМетаданных**, который   * централизованно хранит ссылки на имена объектов метаданных конфигурации, автоматически отслеживает переименования, удаления и добавление объектов метаданных и позволяет избежать массовых операций по замене имен в таблицах; * а также позволяет сократить размер записей таблиц (ссылка вместо строки длиной 255), что улучшает общую производительность системы.   Исключение составляют роли и подсистемы, для которых автоматически не отслеживаются переименования, и для них требуется в явном виде описать переименования. Подробнее см. [документацию к БСП](http://its.1c.ru/db/bspdoc).  В противном случае, если не указать переименование ролей и подсистем, то их ссылки в справочнике **ИдентификаторыОбъектовМетаданных** станут рассогласованными (старый элемент справочника будет помечен на удаление, а вместо него будет создан новый), что приведет к различным ошибкам в тех подсистемах конфигурации, которые опираются на данные этого справочника. Например:   * варианты отчетов, связанные с переименованной подсистемой, исчезнут из панели отчетов; * дополнительные отчеты и обработки, выведенные пользователями в раздел, связанный с переименованной подсистемой, пропадут из списка; * переименованные роли, указанные в профилях групп доступа, не будут назначены пользователям.   3.3. При этом справочник **ИдентификаторыОбъектовМетаданных** не предназначен для хранения ссылок на объекты метаданных других конфигурации (например, в механизмах интеграции с другими системами). Для этих целей рекомендуется использовать строковые реквизиты и реализовывать специальные механизмы по поддержанию актуальности их значений.  3.4. В случаях, когда ведется две и более параллельных «веток» разработки, например, выпускаются версии 2.0 и 3.0 (или организован выпуск исправительных релизов параллельно с выпуском новых функциональных релизов) нужно учесть следующее: в текущей и младших версиях конфигурации запрещается переименование с последующим созданием нового объекта метаданных с тем же полным именем, а также двойное переименование. Правильно выполнять такие переименования только в самой последней версии – 3.0.  В противном случае, при переходе с младшей версии на старшую версию это переименование будет учтено дважды, что приведет к рассогласованию ссылок на объекты метаданных.  При использовании ссылок на справочник **ИдентификаторыОбъектовМетаданных** Библиотеки стандартных подсистем, такой запрет действует только для ролей и подсистем.  Подробнее см. [документацию к БСП](http://its.1c.ru/db/bspdoc) |

### Требования к установке и обновлению прикладных решений

#std731

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Подготовка дистрибутивов установки и обновления прикладных решений (конфигураций) системы 1С:Предприятие 8 должна выполняться в соответствии с рекомендациями, изложенными в [главе 30 «Поставка и поддержка конфигурации»](http://its.1c.ru/db/v83doc#content:88:1) Руководства разработчика из комплекта документации к программным продуктам системы 1С:Предприятие 8.

2. Исходя из данных требований, полный путь к каталогу поставляемых разработчиком шаблонов конфигурации (информационных баз), в общем случае, имеет вид:

<Каталог шаблонов>\<каталог разработчика>\<каталог конфигурации>\<каталог версии>

При этом в каталоге версии вместо точек должны использоваться подчеркивания.  
Например, версия 11.1.3.6 конфигурации «Управление торговлей», редакция 11.1, будет устанавливаться в каталог:

<Каталог шаблонов>\1C\Trade\11\_1\_3\_6\

3. В каталоге версии должен быть расположен файл-манифест 1cv8.mft, в котором описываются установленные шаблоны конфигурации (.cf) и демонстрационные информационные базы (.dt).  
Например, файл-манифест конфигурации «Управление торговлей», редакция 11.1, при установке которой устанавливаются 2 шаблона информационных баз - пустой (рабочей) и демонстрационной базы.

Vendor=Фирма "1С"  
Name=УправлениеТорговлей  
Version=11.1.3.6  
AppVersion=8.3  
[Config1]  
Catalog=1С:Управление торговлей/Управление торговлей  
Destination=1C\Trade  
Source=1cv8.cf  
[Config2]  
Catalog=1С:Управление торговлей/Управление торговлей (демо)  
Destination=1C\DemoTrd  
Source=1Cv8.dt

4.1. В целях исключения совпадения названий конфигураций и возможных в таком случае коллизий при отображении шаблонов информационных баз в диалоге создания ИБ, рекомендуется.

* В строке **Catalog** файла-манифеста, в наименование шаблона конфигурации (часть значения до символа «/») следует включать в явном виде название разработчика. Название разработчика может присутствовать в любом месте наименования шаблона конфигурации, например:  
  **1С:Управление торговлей**или  
  **Управление торговлей (1С)**
* В строке **Destination** рекомендуемый каталог создания информационной базы следует указывать в виде:

Destination = <каталог разработчика>\<каталог информационной базы>

<каталог разработчика> должен содержать название разработчика, но в этом случае - адаптированное для использования в именах файловых каталогов. Например: 1C\Trade

4.2. Для исключения повторений названия разработчиков – поставщиков прикладных решений (конфигураций) должны быть глобально уникальными: название для использования в наименованиях шаблонов конфигураций и название для использования в именах файловых каталогов.

Название разработчика выбирается один раз и в дальнейшем может использоваться разработчиком при установке и регистрации шаблонов всех разработанных им прикладных решений для системы программ 1С:Предприятие 8.

5. Для обновления версий прикладных решений (конфигураций) выпускаются отдельные дистрибутивы обновлений, которые включают в себя файл обновления конфигурации (.cfu).

Основным рекомендуемым способом обновления с помощью конфигуратора является поиск подходящих файлов обновления (меню **Конфигурация – Поддержка – Обновить конфигурацию**). Кроме того, в конфигурациях могут быть предусмотрены и другие средства для обновления версий. При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) такая возможность предусмотрена в подсистеме «Обновление конфигурации».

### Несущественные предупреждения проверки конфигурации

#std759

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Следующие предупреждения проверки конфигурации (конфигуратор – меню **Конфигурация** – **Проверка конфигурации…**) не являются существенными для работоспособности прикладных решений и поэтому не подлежат обязательному исправлению:

* Пустой обработчик (для обработчиков оповещений в программных модулях);
* Неразрешимые ссылки на объекты метаданных (в формах и в справке);
* Неразрешимые ссылки на картинки (в формах);
* Неправильные пути к данным (в формах).

Кроме того методика поиска и исправления подобных мест отсутствует.

2. При регламентной проверке конфигурации не следует включать флажок **Поиск использования синхронных вызовов**, так как в результатах проверки выводятся, в том числе, корректные места вызовов в коде, который не исполняется в веб-клиенте (например, серверный код). См. также: [Ограничение на использование модальных окон и синхронных вызовов](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100703.htm?_=1579516850).

См. также

* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1579516850)

## [Учет версий конфигураций](https://its.1c.ua/db/v8std/browse/13/-1/1/3)

### Общие сведения о выпуске конфигураций

#std484

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Конфигурации выпускаются версиями и редакциями.

2.**Версия** - исправление текущих ошибок и внесение незначительных усовершенствований. Выпуск новой версии должен обеспечивать переход с предыдущей с сохранением данных.

3. **Редакция** - внесение существенных изменений в структуру учета, требующих преобразования данных. Формальным, но не обязательным, признаком новой редакции является необходимость переноса данных путем конвертации. При выпуске новой редакции желательно обеспечивать переход с сохранением данных. Если по каким либо причинам это невозможно, необходимо описать процедуру перехода на новую редакцию (начало работы, перенос начальных остатков и т.д.).

См. также

* [Нумерация редакций и версий](https://its.1c.ua/db/content/v8std/src/100/200/i8100483.htm?_=1579516850)

### Нумерация редакций и версий

#std483

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Номер очередной редакции конфигурации, начинается со следующего целого номера относительно предыдущей редакции. Для обозначения редакции обычно номер редакции объединяют через точку с номером подредакции, например: редакция 1.5, редакция 1.6 и т. д. Для новых конфигураций нумерация начинается с 1.0.

2. Все версии одной подредакции (включая альфа, ознакомительные, бета и финальные версии) нумеруются подряд. Нумерация версий начинается с 1.

3. Информация о номере редакции, номере подредакции и номере версии объединяются в полный номер версии конфигурации. Он указывается в свойстве Версия конфигурации и представляет собой строку символов следующего вида:

{Р|РР}.{П|ПП}.{З|ЗЗ}.{С|СС}

где:

Р - номер редакции (минимум 1 цифра, может занимать и больше разрядов);  
П - номер подредакции (минимум 1 цифра, может занимать и больше разрядов);  
З - номер версии (минимум 1 цифра, может занимать и больше разрядов);  
С - номер сборки (минимум 1 цифра, может занимать и больше разрядов).

Пример:

1.6.4.7 – 7-я сборка, 4-ой версии, редакции 1.6

См. также

* [Общие сведения о выпуске конфигураций](https://its.1c.ua/db/content/v8std/src/100/200/i8100484.htm?_=1579516850)

### Заполнение свойств конфигурации информацией о выпуске

#std482

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. **Синоним**. В синониме указывается официальное название конфигурации, которое будет идентифицировать конфигурацию в документации, на коробке с продуктом, прайс-листе, рекламе, в информационных и методических материалах. В конце официального названия через запятую указывается слово "редакция" и номер редакции.

Например: "Бухгалтерия предприятия, редакция 1.6"

2. **Имя**. Имя образуется по правилам образования имен из синонима; слово "редакция", номер редакции (и подредакции) – не указываются.

Например: "БухгалтерияПредприятия"

3. **Краткая информация**. Краткая информация повторяет синоним.

4. **Подробная информация**. Подробная информация повторяет синоним.

5. **Логотип**. Для типовых конфигураций фирмы "1С" картинка логотипа не заполняется.

6. **Заставка**. В качестве заставки устанавливается картинка одного из типов, поддерживаемых системой 1С:Предприятие. Требования к размеру картинки см. в [документации](https://its.1c.ua/db/v83doc/bookmark/dev/TI000002041).

7. **Авторские права**. Указывается строка вида:

Copyright (C) <разработчик>, хххх-хххх. Все права защищены.

Для конфигураций на английском языке: Copyright (C) <разработчик>, хххх-хххх. All rights reserved.

Вместо "хххх-хххх" указываются конкретные годы выпуска конфигурации.

Например: Copyright (C) ООО "1С-Софт", 2009-2016. Все права защищены

8. **Адрес информации о поставщике**. Для типовых конфигураций фирмы "1С" указывается строка:  
[http://www.1c.ru](http://www.1c.ru/)

9. **Адрес информации о конфигурации**. Для типовых конфигураций фирмы "1С" указывается строка:

[http://v8.1c.ru/<короткое имя>/](http://v8.1c.ru/%3C%D0%BA%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%BE%D0%B5%20%D0%B8%D0%BC%D1%8F%3E/),

где вместо <короткое имя> указывается англоязычное короткое имя конкретной конфигурации, например:

<http://v8.1c.ru/trade/>

10. **Поставщик**. Для типовых конфигураций фирмы "1С"  в качестве поставщика указывается:

Фирма "1С"

11. **Версия**. Указывается полный номер версии конфигурации. Например, 1.6.4.7

Подробнее об образовании номера версии см. раздел [Нумерация редакций и версий](https://its.1c.ua/db/content/v8std/src/100/200/i8100483.htm?_=1579516850).

12. **Адрес каталога обновлений**. Для типовых конфигураций фирмы "1С" указывается строка:

<http://downloads.v8.1c.ru/tmplts/>

## [Организация хранения данных](https://its.1c.ua/db/v8std/browse/13/-1/1/4)

### Общие сведения об организации хранения данных

#std683

Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При проектировании системы одной из задач является выбор типов объектов метаданых для реализации хранения соответствующих сущностей предметной области. Неправильный выбор типов объектов ведет к неэффективности прикладного решения, невозможности его последующего развития и делает невозможным адаптацию к возможным изменениям состава решаемых задач.

2. При выборе типа объектов метаданных в общем случае следует руководствоваться следующей схемой прикладного решения:



\* Стрелки на схеме обозначают взаимосвязи между данными (взаимные ссылки).

На схеме выделяются следующие блоки:

1. **Условно-постоянная информация**. К этой части относится информация, которая вводится один раз, сравнительно редко изменяется и многократно используется. Примером такой информации могут служить различные классификаторы, настройки, перечни, реестры, нормативно-справочная информация и т.п.

2. Различные **события процессов предметной области**, которые привязаны ко времени и могут порождать при регистрации различные сведения, изменять значения показателей. Пример – документооборот предприятия, ведение учета; регистрация заявок, звонков и т. п.

3. **Накопленные сведения, значения показателей**, которые характеризуют процессы и текущее состояние прикладной области. В отличие от первых двух частей, эти данные имеют необъектную природу и не являются самостоятельными сущностями с точки зрения прикладной области. Пример – история продаж товара, остатки на складах, текущий бухгалтерский баланс, история изменения курсов валют и т.п.

Отдельно выделяются средства для анализа и обработки данных, отчеты, механизмы, которые опираются на данные всех остальных блоков, но сами данных не содержат.

Подробнее о задачах и принципах хранении информации см. в книге «[Профессиональная разработка в системе 1С Предприятие 8](https://buh.ru/books/detail.php?ID=42696)» глава 6.

2.1. Таким образом, упрощенно, для каждой сущности предметной области необходимо выбрать соответствующий блок по следующей схеме:

* Если необходимо хранить сравнительно редко изменяющуюся информацию, не привязанную ко времени, то это блок условно постоянной информации (1).
* Если необходимо регистрировать события, требующие документально подтверждения с отслеживанием последовательности событий на временной оси, то это блок событий процессов предметной области (2).
* В противном случае сущность должна относиться к блоку сведений и значений показателей (3).

Более подробные критерии выбора того или иного блока:

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий / блок** | **Условно-постоянная информация** | **События процессов предметной области** | **Накопленные сведения, значения показателей** |
| Основное предназначение | Необходимо хранить нормативно-справочную информацию, реестры | Необходимо регистрировать события процессов, обеспечивать документальное подтверждение сведениям | Необходимо хранить данные, которые характеризуют процессы и текущее состояние прикладной области |
| Отслеживание изменения состояния | Не требуется | Требуется регистрация документа к учету, отмена регистрации учета документа, учет запусков или окончаний процесса, изменения состояния задач, формирование движений | Не требуется |
| Иерархия, группировка данных | Требуется иерархия и группировка, возможно между разными сущностями | Не требуется | Не требуется |
| Ключевые свойства | Необходимо иметь наименование, код | Необходимо учитывать дату события, его номер | Не требуется |
| Хранение значений дополнительных реквизитов сущности | Необходимо хранить редко изменяемые реквизиты произвольных данных | Необходимо хранить ссылки на другие объекты и значения параметров, характеризующие событие | Необходимо хранить только значения реквизитов для других объектов базы |
| Нумерация | Необходимы серии кодов по всем элементам данного типа или в пределах иерархии | Необходимы серии номеров по всем элементам данного типа или в пределах периода по дате, сквозная нумерация объектов разных типов | Не требуется |

2.2. Затем, необходимо принять решение о конкретном виде типа объекта метаданных внутри выбранного блока:

2.2.1. Для хранения условно-постоянной информации:

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.   * 1. Если требуется хранение плана счетов для организации учета по [принципам двойной записи](http://its.1c.ru/db/garant#content:10036812:1:1), то используется специализированный объект метаданных «[План счетов](http://its.1c.ru/db/v8devgloss#content:28:hdoc:01)»   2. Если требуется хранение перечня видов расчета для организации [учета начислений и удержаний](http://its.1c.ru/db/garant#content:12025268:1:1020), то используется специализированный объект метаданных «[План видов расчета](http://its.1c.ru/db/v8devgloss#content:28:1:03)»   3. Если требуется хранить список характеристик (свойств), причем состав самого списка, тип характеристик, их состав определяются пользователем, то используется объект метаданных «[План видов характеристик](http://its.1c.ru/db/v8devgloss#content:28:hdoc:02)» |

4. Если требуется хранить одиночное значение, которое редактирует пользователь (как правило, это администратор, выполняющий настройки системы), не требующего ссылок из других данных, то используется объект метаданных «[Константа](http://its.1c.ru/db/v8devgloss#content:26:hdoc)»  
5. Если необходимо определить фиксированный список значений не редактируемый пользователем, без каких-либо дополнительных реквизитов, то используется объект метаданных «[Перечисление](http://its.1c.ru/db/v8devgloss#content:62:hdoc)»  
6. В остальных случаях, как правило, используется объект метаданных «[Справочник](http://its.1c.ru/db/v8devgloss#content:22:hdoc)»

Более подробные критерии выбора того или иного вида объекта метаданных:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий / тип объекта** | **Константа** | **Перечисление** | **План видов характеристик \*** | **Справочник** |
| Основное предназначение | Необходимо хранение одиночных значений, предопределенных данных | Необходимо хранение списка неизменных представлений без дополнительных их атрибутов | Необходимо хранение списка сущностей и значений характеристик экземпляров сущности | Требуется хранение списка объектов и значений их атрибутов |
| Добавление и редактирование пользователем | Требуется только изменение значения | Не требуется | Требуется добавление, удаление и изменение элементов, редактирование состава и значений характеристик сущности | Требуется добавление, удаление, изменение элементов |
| Иерархия, группировка данных | Не требуется | Не требуется | Требуется в пределах одной сущности | Требуется в пределах одной сущности или между разными сущностями |
| Хранение значений дополнительных реквизитов сущности | Не требуется | Не требуется | Необходимо хранить произвольные данные для атрибутов сущности | Необходимо хранить произвольные данные для атрибутов сущности |
| Хранение списков значений дополнительных реквизитов | Не требуется | Не требуется | Требуется хранение списков наборов значений реквизитов для сущности | Требуется хранение списков наборов значений реквизитов для сущности |
| Возможность ввода на основании других объектов | Не требуется | Не требуется | Необходим ввод новых элементов с использованием информации других объектов | Необходим ввод новых элементов с использованием информации других объектов |
| Нумерация | Не требуется | Не требуется | Необходимы серии кодов по всем элементам одного типа или в пределах группировки | Необходимы серии кодов по всем элементам одного типа или в пределах группировки или подчинения |

\* область применения (уточнение): управляемое приложение, обычное приложение.

2.2.2. Для хранения событий процессов предметной области:

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1. Если требуется учет одиночных событий, адресованных некоторому исполнителю (пользователю, сотруднику, группе или роли) не требуется формирование движений по результатам события: то используется объект метаданных «[Задача](http://its.1c.ru/db/v8devgloss#content:94:hdoc)» 2. Если требуется регистрировать в системе возникновение и ход регулярного процесса, состоящего из последовательности действий (событий), то используется объект метаданных «[Бизнес-процесс](http://its.1c.ru/db/v8devgloss#content:42:hdoc)». Для учета событий, действий в рамках процесса используется объект метаданных «[Задача](http://its.1c.ru/db/v8devgloss#content:94:hdoc)» |

3. В остальных случаях, как правило, используется объект метаданных «[Документ](http://its.1c.ru/db/v8devgloss#content:23:hdoc)»

Более подробные критерии выбора того или иного вида объекта метаданных:

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий / тип объекта** | **Задача \*** | **Бизнес-процесс \* (с задачами)** | **Документ** |
| Основное предназначение | Необходимо вести учет одиночных событий, адресованных некоторым исполнителям | Необходимо вести учет последовательности событий, адресованных некоторым исполнителям | Необходима регистрация событий во времени, генерация вторичных данных, соответствующих этим событиям |
| Вложенность | Не требуется | Требуется учет процессов, вложенных в другие процессы (иерархия задач) | Не требуется |
| Объединение в журналы | Не требуется | Не требуется | Необходимо объединение документов разных видов в одном журнале |
| Состояние объекта | Требуются состояния «новый», «выполнено» | Требуются состояния «новый», «в работе», «завершен» | Требуются состояния «проведен», «не проведен» |
| Нумерация | Необходимы серии номеров по всем задачам данного вида или в пределах периода по дате | Необходимы серии номеров по всем процессам данного вида или в пределах периода по дате, нумерация событий внутри процесса | Необходимы серии номеров для документов разных видов – сквозные, или в пределах периода по дате |

\* область применения (уточнение): управляемое приложение, обычное приложение.

2.2.3. Для хранения накопленных сведений, значений показателей:

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1. Если требуется хранение данных учета с использованием [принципа двойной записи](http://its.1c.ru/db/garant#content:10036812:1:1), то используется специализированный объект метаданных «[Регистр бухгалтерии](http://its.1c.ru/db/v8devgloss#content:20:hdoc:04)» 2. Если требуется хранение результатов расчета [учета начислений и удержаний](http://its.1c.ru/db/garant#content:12025268:1:1020), то используется специализированный  объект метаданных «[Регистр расчета](http://its.1c.ru/db/v8devgloss#content:20:hdoc:05)» |

3. Если требуется хранение изменений показателей – приход и расход, получение остатков и оборотов за период, то используется объект метаданных «[Регистр накопления](http://its.1c.ru/db/v8devgloss#content:151:hdoc)».  
4. Во всех остальных случаях используется объект метаданных «[Регистр сведений](http://its.1c.ru/db/v8devgloss#content:152:hdoc)».

Более подробные критерии выбора того или иного вида объекта метаданных:

|  |  |  |
| --- | --- | --- |
| **Критерий / тип объекта** | **Регистр накоплений** | **Регистр сведений** |
| Основное предназначение | Требуется хранение изменений данных - прихода и расхода значений показателей | Требуется хранение информации в виде наборов записей, регистрация некоторых сведений, значений |
| Получение данных | Требуется получение остатков, оборотов данных | Необходимо получение среза информации на момент времени или текущего значения показателей |
| Подтверждение происхождения данных | Требуется обязательная связь с регистрирующим документом | Связь не обязательна |

3. Пример выбора типов объектов метаданных.  
Пусть некоторая организация занимается периодическим анкетированием. При заполнении анкеты указывается дата анкетирования. В анкете указывается набор вопросов, результатом заполнения анкеты является набор ответов. Сущность «Анкета» привязана к дате, порождает статистику – ответы на вопросы.

Таким образом, имеем:

* Периодическое событие, привязано к дате, порождает значение параметров. Это - второй блок «события предметной области».
* Далее уточняем внутри блока «событий предметной области»: т.к. анкета формирует вторичные данные - результаты ответов на вопросы, то следовательно, это должен быть документ.

### Уточнение сущности объекта метаданных

#std541

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При описании структуры прикладного решения следует использовать разные типы объектов метаданных для реализации разных по смыслу сущностей. Например, сущность «организации» описывается справочником **Организации**, а сущность «подразделение» - отдельным справочником **Подразделения**.

2. В то же время для уточнения сущности того или иного объекта, у объекта могут быть заведены реквизиты, в зависимости от значения которых меняется его смысловая нагрузка и поведение. Например, в справочнике **Организации** могут одновременно храниться «обычные» организации и обособленные подразделения (т.е. организации, у которых имеется головная организация), а в справочнике **Сотрудники** – принятые на работу и уволенные сотрудники.

В этих случаях рекомендуется заводить отдельные реквизиты, которые бы однозначно определяли вид или состояние объекта. Это могут быть реквизиты типа **Булево** или перечисления с видами объекта. Не следует трактовать вид объекта по косвенным признакам, в частности в зависимости от заполнения того или иного реквизита.

Например, неправильно

* заводить в справочнике **Организации** реквизит **ГоловнаяОрганизация**, в зависимости от заполнения которого прикладная логика трактует вид организации – «обычная» или обособленное подразделение;

правильно:

* помимо реквизита **ГоловнаяОрганизация**, завести в справочнике **Организации** булев реквизит **ОбособленноеПодразделение**, в зависимости от значения которого (**Истина** или **Ложь**) однозначно трактуется вид организации, и в частности необходимость заполнения реквизита **ГоловнаяОрганизация**.

3. В то же время, если например, у справочника **Сотрудники** имеются реквизиты **ДатаПриема** и **ДатаУвольнения**, то введение пары реквизитов булево типа **ПринятНаРаботу** и **Уволен** не оправдано. Для подобных сущностей с несколькими состояниями целесообразно завести один реквизит типа перечисление со статусами, например: «работает», «уволен», состав которых при необходимости может быть расширен и другими значениями.

### Имя, синоним, комментарий

#std474

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. **Синоним** объекта должен быть определен так, чтобы осмысленно, лаконично описывать объект. Заполняется обязательно.

Данное требование продиктовано тем, что синонимы непосредственно участвуют в формировании пользовательского интерфейса (отображаются в формах, отчетах, командном интерфейсе и т.д.) и поэтому должны корректно и одинаково во всех местах пользовательского интерфейса идентифицировать ту сущность, к которой они относятся. Помимо объектов метаданных, требование распостраняется также и на реквизиты объектов метаданных, табличные части, реквизиты табличных частей, измерения регистров, ресурсы и другие объекты конфигурации, у которых имеется синоним.

1.2. Не рекомендуется в синонимах объектов использовать сокращения. Исключением являются только общеупотребительные и соответствующие целевой аудитории сокращения (например, **Сумма (регл.)** ) и аббревиатуры (например, **НДС** или **МСФО**).

1.3. В синонимах объектов и текстовых сообщениях пользователю должны использоваться общепринятые термины, понятные пользователю. Не должно быть сленга, искажения названий продуктов и компаний; англоязычных фраз, записанных русскими буквами; русскоязычных английскими буквами и т.п.

В частности, если для англоязычного термина нет общепринятого перевода на русский язык, то следует использовать оригинальный англоязычный термин.  
Например, неправильно: «Загрузка данных из Эксель», «Загрузка данных из MS Excel»,  
правильно: «Загрузка данных из Microsoft Excel».

1.4. В случае если у объекта метаданных имеются стандартные реквизиты (стандартные табличные части), для них также следует указывать синонимы, исходя из прикладного смысла каждого реквизита.

1.5. При этом для стандартных реквизитов **Родитель** и **Владелец**, следует всегда указывать синонимы, отличные от синонимов по умолчанию. Например, в конфигурации имеется справочник **Файлы** со стандартным реквизитом **Владелец** типа **СправочникСсылка.ПапкиФайлов**. В этом случае  
неправильно

* оставлять синоним стандартного реквизита **Владелец** по умолчанию: «Владелец»;

правильно

* вложить в синоним прикладной смысл: «Папка» или «Папка с файлом».

Другой пример. В то время как для стандартного реквизита **Наименование** некоторых справочников может вполне подойти синоним по умолчанию «Наименование», в случае со справочником **Файлы** целесообразнее назначить синоним «Имя файла», а для справочника **ФизическиеЛица** – дать синоним «ФИО».

См. также: [*Пользовательские представления объектов*](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/i8100468.htm?_=1579516850), [*Тексты*](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/1%C2%A0800/i8100598.htm?_=1579516850)

1.6. В случае, когда есть два (или более) объекта метаданных со схожим назначением, необходимо, чтобы синонимы каждого объекта полностью описывали каждый объект.

Например, неправильно давать справочникам следующие синонимы:

* **Банковские счета**,
* **Банковские счета контрагентов**

правильно:

* **Банковские счета организаций**,
* и **Банковские счета контрагентов**

Следует называть эти объекты явным образом, чтобы пользователь не задавался вопросом: «Если в справочнике ***Банковские счета контрагентов*** хранится информация о счетах контрагентов, то информация о чьих счетах хранится справочнике ***Банковские счета***?»

Это требование справедливо и для синонимов подчиненных объектов метаданных (реквизитов, табличных частей, измерений, ресурсов и пр.).

Пример с реквизитами табличной части **«Товары»** документа **«Пересчет товаров»**.

Неправильно:

* **Количество**
* **Количество (по учету)**

правильно:

* **Количество (в наличии)**
* **Количество (по учету)**

Пример со стандартным реквизитом **Наименование** и еще одним реквизитом справочника **«Номенклатура»**.  
Неправильно:

* **Наименование**
* **Полное наименование**

правильно:

* **Рабочее наименование**
* **Наименование для печати**

2.1. **Имя** объекта рекомендуется строить на основе синонима: пробелы и пр. недопустимые в имени символы, удаляются, а первые буквы слов делаются прописными.

Например, неправильно:

* у справочника **НаборыДополнительныхРеквизитовИСведений** задан синоним «Дополнительные свойства»
* у общей команды **ПрисоединенныеФайлы** – синоним «Файлы присоединенные»;

правильно:

* у справочника **НаборыДополнительныхРеквизитовИСведений** задан синоним «Наборы дополнительных реквизитов и сведений»
* у общей команды **ПрисоединенныеФайлы** – синоним «Присоединенные файлы».

Также допустимы ситуации, когда имя более кратко описывает объект, чем синоним – когда в имени «сокращены» одно или несколько последних «малозначащих» слов из синонима. Например:

* **ДлительностьОжиданияСервера** – синоним «Длительность ожидания сервера (сек)»
* **КоличествоЕдиниц** – синоним «Количество единиц измерения»
* **Обработки.ГрупповоеИзменениеОбъектов.Операции.ИмяРеквизита** – синоним «Имя реквизита (свойство)»

Имя также может не включать союзы и предлоги из текста синонима, например: для реквизита **ЗначениеСкидкиНаценки** синоним «Значение скидки или наценки».

А также наоборот, допустимы ситуации, когда синоним более кратко описывает объект, чем имя – когда в синониме «сокращены» одно или несколько последних «технических» слов из имени.

Данное требование продиктовано тем соображением, что объекты конфигурации и их представления в пользовательском интерфейсе должны быть максимально легко узнаваемыми, например, на внедрении, которое проводится не самими разработчиками конфигурации, а силами технических специалистов по внедрению.

См. также: [Дополнительные требования по именам объектов метаданных в конфигурациях](https://its.1c.ua/db/content/v8std/src/100/100/i8100550.htm?_=1579516850)

2.2. Исключение из этого правила составляют [объекты метаданных с префиксом **Удалить**](https://its.1c.ua/db/content/v8std/src/100/300/i8100534.htm?_=1579516850).

Также не следует переименовывать объекты метаданных (их реквизиты, формы и пр.) при пересмотре их синонимов, если на эти объекты распространяются требования обеспечения обратной совместимости. В частности, при разработке библиотек необходимо [обеспечивать обратную совместимость между различными версиями библиотек в пределах одной редакции библиотеки](https://its.1c.ua/db/content/v8std/src/800/i8100644.htm?_=1579516850).

2.3. Имена объектов метаданных не должны превышать 80 символов.

2.4. Для подчиненных объектов метаданных, таких как реквизиты, измерения, ресурсы рекомендуется не использовать имена, совпадающие с именами объектов-владельцев. Например, измерение **Пользователь** (типа **СправочникСсылка.Пользователи**) регистра сведений **ИсполнителиЗадач** названо некорректно; правильное название измерения, раскрывающее его смысл: **Исполнитель**.

2.5. Также рекомендуется не использовать имена, которые применяются при именовании таблиц языка запросов (например, **Документ**, **Справочник**, **РегистрСведений** и т.д.). Такие имена могут приводить к ошибкам при исполнении запроса, затрудняют использование конструктора запроса и снижают наглядность текста запроса. Например, выполнение данного запроса вызывает ошибку:

ВЫБРАТЬ  
Сведения.Сведения  
ИЗ  
РегистрСведений.Сведения КАК Сведения

3.1. **Комментарий** задается только в тех случаях, когда необходимо дать участнику разработки конфигурации какие-либо пояснения по данному объекту конфигурации. Например, комментарий к реквизиту справочника может быть таким: "Индексирование поставлено для оптимизации отчетов с отбором по виду контрагента", или: "Используется в регламентированном учете".

3.2. Комментарий начинается с прописной буквы, точки ставятся только после сокращений.

4. В именах, синонимах и комментариях не допускается использовать букву "ё".

См. также

* [Пользовательские представления объектов](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/i8100468.htm?_=1579516850)
* [Подсказка и проверка заполнения](https://its.1c.ua/db/content/v8std/src/100/300/i8100478.htm?_=1579516850)
* [Реквизит "Комментарий" у документов](https://its.1c.ua/db/content/v8std/src/100/300/i8100531.htm?_=1579516850)

### Подсказка и проверка заполнения

#std478

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1.1. **Свойство «Подсказка»**. Задается для тех объектов (реквизитов объектов, реквизитов табличных частей, измерений и ресурсов регистров), которые выводятся пользователю в виде элементов интерфейса и которые требуют пояснения, расшифровки и донесения до пользователя подробного описания их назначения. Для реквизитов, используемых в ежедневной работе, подсказки добавлять не следует.  В тексте подсказки не рекомендуется приводить инструкции и объяснять работу реквизита, вместо этого следует оптимизировать сценарии работы, с ним связанные.  При этом бессмысленные подсказки, автоматические генерируемые конфигуратором при создании элементов управления (групп) на формах, нужно не забывать очищать. Например, «Группа шапка» и пр.  См. также: [Подсказки на форме](https://its.1c.ua/db/content/v8std/src/1%C2%A0000/1%C2%A0400/i8100721.htm?_=1579516850), [Подсказки для интерфейсных подсистем](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/400/i8100571.htm?_=1579516850), [Тексты](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/1%C2%A0800/i8100598.htm?_=1579516850) |

2.1. **Свойство «Проверка заполнения».** Для всех типизированных объектов метаданных, а также для стандартных реквизитов и табличных частей, которые в соответствии с логикой объекта являются обязательными к заполнению, свойство "Проверка заполнения" должно быть установлено в "Выдавать ошибку".

В ряде случаев проведение документа с незаполненными реквизитами и табличными частями не имеет смысла с точки зрения отражения документа в учете. Например, документ **Заказ клиента** является запросом клиента на поставку определенного количества товара. Из определения понятно, что методически заказ с незаполненным клиентом и незаполненной табличной частью **Товары** не имеет смысла, поэтому у реквизита **Клиент** и табличной части **Товары** свойство "Проверка заполнения" должно быть установлено в "Выдавать ошибку".

2.2. При установке свойства «Проверка заполнения» следует исходить из того, что все ограничения и проверки должны быть (насколько это возможно полно) описаны в метаданных конфигурации. Поэтому если хотя бы один из сценариев работы с объектом требует обязательного заполнения реквизита, то свойство «Проверка заполнения» устанавливается в «Выдавать ошибку». Если в других сценариях работы заполнять реквизит не обязательно, то такие случаи должны быть предусмотрены в [обработчике события модуля объекта **ОбработкаПроверкиЗаполнения**](https://its.1c.ua/db/content/v8std/src/100/400/i8100463.htm?_=1579516850).

При этом не следует придерживаться обратной схемы, когда свойство «Проверка заполнения» установлено в «Не проверять», а в обработчике **ОбработкаПроверкиЗаполнения** дописаны какие-либо проверки заполнения. Такая схема затрудняет анализ логики работы конфигурации.

2.3. Если проверка заполнения реквизита зависит от тех или иных условий, рекомендуется управлять автопометкой незаполненного значения с помощью условного оформления форм объектов. Убирать ее в случае, если при данном состоянии объекта заполнение реквизита проверять не требуется.

См. также

* [Имя, синоним, комментарий](https://its.1c.ua/db/content/v8std/src/100/300/i8100474.htm?_=1579516850)

### Использование кодов (номеров) объектов конфигурации

#std473

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  Длины кодов (номеров) объектов конфигурации задаются в зависимости от их прикладного смысла.   1. Необходимость применения кодов (номеров) объектов конфигурации определяется из их прикладного смысла. Основания для применения кодов (номеров):   1.1. Пользователям предстоит работать со списками, содержащими большое количество элементов. Пример: справочник **Номенклатура**.  1.2. Искать данные по коду (номеру) удобнее, чем по другим свойствам. Например, справочник **Статьи расходов**. Пользователю проще запомнить код, чем каждый раз вчитываться в название статьи. Кроме того, названия статей могут меняться.  1.3. Код имеет прикладной смысл, продиктованный спецификой автоматизируемой области. Например, код справочника **Единицы измерения** соответствует коду единицы измерения в ОКЕИ, код справочника **Номера ГТД** соответствует номеру грузовой таможенной декларации и т.д.   1. Необходимость применения автонумерации объектов конфигурации отсутствует в следующих случаях:   2.1. Код используется как краткое представление элемента данных в виде строки. Например, для справочника **Пользователи** автонумерация не применяется, т.к. в коде хранится краткое имя пользователя (логин).  2.2. Код (номер) получается из внешних источников (т.н. входящие данные). Например, это все классификаторы. Другой пример - справочник **Номера ГТД**, код которого вводится исходя из данных входящих документов.   1. Длина кода (номера) устанавливается в зависимости от его прикладного назначения и метода получения (ввода):   3.1. Для объектов с автонумерацией длина кода (номера) выбирается, исходя из потенциального количества всех объектов, хранимых в базе данных; объектов, относящихся к определенному периоду (для документов и бизнес-процессов); или объектов, относящихся к определенному владельцу (иерархические и подчиненные справочники, задачи). При этом в длине номера необходимо учитывать длину префиксов нумерации, например, префикс информационной базы, префикс организации, если это предусмотрено конфигурацией и т.п.  При разработке типовых конфигураций рекомендуемыми, но не обязательными к применению являются длины кодов (номеров) из следующего ряда: 3, 5, 9, 11. При этом в длине номера необходимо учитывать длину префиксов нумерации, например, префикс информационной базы, префикс организации, если это предусмотрено конфигурацией и т.п.  Если в конфигурации используется подсистема **Префиксация объектов** из **Библиотеки стандартных подсистем**, то совокупную длину (с учетом префикса) номеров документов и кодов справочников рекомендуется устанавливать не менее 11 символов (11, 13, 15, …). Подробнее см. [документацию к подсистеме "Префиксация объектов" на ИТС](https://its.1c.ua/db/bspdoc).  3.2. Для объектов, в которых код используется как краткое представление элемента данных в виде строки (см. п. 2.1) длина кода устанавливается достаточной для хранения краткого строкового представления объектов исходя из прикладного смысла кода.  3.3. Для объектов, в которых код (номер) получается из внешних источников (см. п. 2.2), длина кода (номера) зависит от этого источника.  3.4. Рекомендуется устанавливать допустимую длину кода (номера) объектов переменной.   1. В случае если прикладное решение рассчитано на работу с данными, которые могут вводиться параллельно из нескольких мест (в рамках РИБ, в других программах), в нем должна быть реализована возможность автоматической префиксации объектов конфигурации, для которых выполняются следующие условия: - используется строковый код (номер), - используется автонумерация, - данные, соответствующие области, в пределах которой коды (номера) должны быть уникальными, могут вводиться параллельно из нескольких мест (узлов РИБ, программ) и впоследствии консолидироваться, например, в результате выполнения синхронизации данных. Пример такой области для большинства видов документов – организация и период. При использовании в конфигурации **Библиотеки стандартных подсистем** реализовать данное требование позволяет подсистема **Префиксация объектов**. |

### Использование реквизитов строкового типа

#std432

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Для реквизитов строкового типа следует использовать переменную длину строки (свойство **Допустимая длина** = **Переменная**) и при этом указывать максимально допустимую длину строки. Свойство **Допустимая длина** может принимать значение **Фиксированная** только в тех случаях, когда при манипуляции этими данными действительно необходимо иметь гарантию, что строка имеет определенную длину (за счет автоматического дополнения концевыми пробелами).

1.2. В тех случаях, когда максимальная длина строки заранее известна (например, она регламентирована), следует указывать ее в свойстве **Длина** (или **Длина наименования** для стандартного реквизита **Наименование**). Например, длина строкового реквизита **ИНН** справочника **ФизическиеЛица** должна составлять 12 символов.

1.3. Если строка является конкатенацией других строк, то ее длина может быть определена как сумма длин исходных строк. Например, длина представления адреса должно равняться сумме длин полей, в которых хранятся части адреса.

1.4 Если длина строки не регламентирована, то рекомендуется выбирать такую длину, которой достаточно для хранения данных в большинстве случаев. Например, для хранения полного наименования контрагента в подавляющем большинстве случаев достаточно 250 символов, максимальная длина имени файла в большинстве файловых систем – 260, полное имя физического лица – 100 и т.п.

2. В отдельных случаях, допускается использование строк неограниченной длины:

2.1. Предполагается, что в реквизит строкового типа может быть помещен пользовательский текст, объем которого может быть значительным. Как правило, это многострочные поля на форме. Например, в поле **Дополнительное описание** в заказе клиента менеджер может поместить всю историю переписки с клиентом, в [поле **Комментарий**](https://its.1c.ua/db/content/v8std/src/100/300/i8100531.htm?_=1579516850) – пользователь может ввести произвольный многострочный текст и т.п.

2.2. В строковом реквизите хранится различная техническая информация, которая генерируется программами и, чаще всего, не предназначена для чтения пользователем, а используется в различных алгоритмах обработки информации. Например, xml-документы, заголовки почтовых сообщений и т.п.

3. В случае использования строковых реквизитов неограниченной длины следует иметь в виду возникающие при этом ограничения в языке запросов:

3.1. При необходимости сравнения значений, группировки и получения различных, такие реквизиты необходимо выражать как строку определенной длины, такой, чтобы выражение было вычислено верно.

Для этих целей в запросе рекомендуется использовать конструкцию

ВЫРАЗИТЬ КАК СТРОКА(1000)

3.2. В отчетах СКД для таких полей следует, вместо этого, задавать параметр **Тип** **значения**поля (на закладке **Наборы данных**).

Следует иметь в виду, что частое приведение неограниченной строки к определенной длине в запросах и отчетах СКД может быть признаком неправильного проектного решения и служить сигналом для пересмотра типа строкового реквизита в пользу ограниченной длины строки.

3.3. В остальных случаях, урезать строку в запросах не требуется.

4. Если в печатных формах предусмотрено отображение строкового поля, то независимо от того, какая назначена длина строки, необходимо обеспечить вывод таких строк полностью, без обрезания части строки. В противном случае, может быть утеряна значимая часть информации. Например, номер дома и квартиры в поле с адресом доставки товара в печатной форме.

Для быстрого выявления в конфигурации всех строковых реквизитов неограниченной длины можно воспользоваться приложенной обработкой [**СтрокиНеограниченнойДлины.erf**](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%B8%D0%9D%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9%D0%94%D0%BB%D0%B8%D0%BD%D1%8B/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%B8%D0%9D%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9%D0%94%D0%BB%D0%B8%D0%BD%D1%8B.zip)

См. также

* [Формирование печатных форм](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/i8100548.htm?_=1579516850)

### Ограничения на использование реквизитов составного типа

#std728

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1.1. Реквизиты составного типа, используемые в условиях соединений, отборах, а также для упорядочивания, должны содержать только ссылочные типы (**СправочникСсылка.[…]**, **ДокументСсылка.[…]** и пр.). В состав их типов не рекомендуется включать никаких других нессылочных типов, например: **Строка**, **Число**, **Дата**, **УникальныйИдентификатор**, **Булево**, а также **ХранилищеЗначения**.  В противном случае производительность запросов заметно снизится. Это обусловлено особенностями физического хранения реквизитов составных типов в колонках таблиц СУБД. См. [Особенности хранения составных типов данных](http://its.1c.ru/db/metod81#content:1828:1) (статья на ИТС).  1.2. В отдельных случаях, для выполнения этой рекомендации можно применить следующий подход.  Например, в документе определен реквизит **Адрес** составного типа, который включает ссылку на справочник **Контакты** и «строку» для возможности ввода в реквизит произвольных строковых значений. Вместо этого, следует предусмотреть отдельный справочник **ПроизвольныеАдреса** и указать его в реквизите **Адрес** вместо строки. При этом добавление новых элементов в справочник **ПроизвольныеАдреса** следует обеспечить автоматически, «незаметно» от пользователя, в момент записи документа. А удаление ненужных элементов справочника **ПроизвольныеАдреса** можно организовать посредством регламентного задания. |

2.1. Для типизированных объектов метаданных, хранящихся в информационной базе, не следует использовать составные типы **ЛюбаяСсылка**, **СправочникСсылка**, **ДокументСсылка** и аналогичные. Состав типов того или иного типизированного объекта должен определяться явным образом.

Исключение составляют универсальные механизмы (алгоритмы), рассчитанные на работу с произвольными ссылочными объектами.

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  2.2. В случае если составной тип массово используется в объектах какой-либо подсистемы или во всей конфигурации, то рекомендуется использовать [определяемые типы](https://its.1c.ua/db/content/v8std/src/100/100/i8100704.htm?_=1579516850). |

См. также:

* [Разыменование ссылочных полей составного типа в языке запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100654.htm?_=1579516850)

### Требования к проведению документов

#std603

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Документы предназначены для ввода первичной информации, связанной с регистрацией событий, воздействующих на учитываемые в системе показатели. Например, при автоматизации финансово-хозяйственной деятельности предприятия – это учет различных хозяйственных операций; в системах управления производственными процессами – регистрация производственных операций и т. д.

2.1. Регистрация события в системе (т.е. отражение его в учете) выполняется с помощью проведения документа. Большинство документов должны проводиться (свойство **Проведение** установлено в значение **Разрешить**).

Логически, непроведенный документ отличается от проведенного тем, что непроведенный документ является «черновиком», не отраженным в учете. Такие документы могут быть сохранены в системе, даже если они не полностью или вообще не заполнены; к ним не применяются никакие проверки и ограничения бизнес-логики (проверки заполнения, дат запрета изменения и т.п.). Данные таких документов не отражаются в учете (не выводятся в отчетах и т. п.)

В то же время, проведенный документ – это «чистовик», формирование и обработка которого завершены и по поводу которого принято решение, что данный документ должен участвовать в учете.

2.2. Если жизненный цикл документа состоит из нескольких этапов, которые соответствуют этапам некоторого процесса, то для описания этих этапов у документа могут быть введены дополнительные статусы. Например, документ **Заказ клиента** может иметь статусы: **Не согласован**, **К обеспечению**, **Закрыт**; документ **Расходный кассовый ордер** – сначала зарегистрирован в журнале регистрации кассовых ордеров (**КО-3**), затем подписан главным бухгалтером (руководителем), передан в кассу, затем зарегистрирован в Кассовой книге, подписан главным бухгалтером (руководителем).

В таких случаях, проведение документа соответствует моменту первичного отражения события в учете, а статусы проведенного документа уточняют, как именно событие отражено в учете.

Если документ проведен, то при переводе документа между статусами пользователям может быть предложено дозаполнить определенные данные документа, к этим данным могут быть применены определенные проверки и ограничения бизнес-логики, специфичные для каждого этапа. До момента проведения, перевод «черновика» документа по статусам не контролируется системой.

Примеры поведения документов с многоэтапным отражением в учете:

* для проведенного документа **Заказ клиента**:
  + при переводе в статус **Не согласован** система контролирует только основные параметры заказа;
  + при переводе в статус **К обеспечению** – обязательно для заполнения поле **Дата отгрузки**, так как логисту необходима информация, к какой дате нужно привезти заказ;
* для проведенного документа **Расходный кассовый ордер** перевод в финальный статус **Зарегистрирован в Кассовой книге и подписан главным бухгалтером (руководителем)** означает, что система должна создать бухгалтерские записи, а отчет кассира будет зарегистрирован в журнале-ордере (или другом регистре учета, например, в бюджетных организациях - в журнале операций).

2.3. Исключение из этого правила («большинство документов должны проводиться») составляют

* документы, которые не предназначены для отражения событий в учете. С помощью таких документов только регистрируют различные события с привязкой ко времени: например, входящую корреспонденцию, звонки, встречи и т.п.
* отдельные документы, технология проведения которых сильно отличается от технологических возможностей платформы, но которые должны выглядеть для пользователя так, как будто они проводятся. Например, это документы **Операция (бухгалтерский и налоговый учет)** – для ввода операций вручную, **Регламентная операция** – для выполнения операции закрытия месяца с возможностью ручной корректировки движений и т.п.

Такие документы не проводятся.

2.4. В случае если пользователь должен выполнять регистрацию события в системе и отражение его в учете за одно действие, необходимо записывать новый документ в режиме проведения.

При этом недопустимо решать эту задачу другими способами, в частности, с помощью отключения проведения у документа.

3. При отражении события в учете может возникнуть необходимость сформировать «вторичные» данные, со сложными привязками к моментам времени, периодам и к другим объектам системы. В этом случае следует помещать такие данные в регистры. Формирование движений по регистрам следует выполнять при проведении: автоматически или вручную.

При автоматическом формировании движений, пользователь вводит информацию о событии в данные документа, а при проведении на основе введенной в документ информации генерируются движения в различные регистры. Например, для бухгалтерских операций происходит формирование проводок.

При ручном формировании движений, пользователь вводит данные непосредственно в регистры. Такие документы обычно называются ручными операциями. Они могут использоваться для введения начальных остатков, или для ввода хозяйственных операций, которые не были предусмотрены разработчиком конфигурации.

4. В отдельных случаях, формирование движений может выполняться отдельным документом. Это востребовано в случае схожей обработки разных видов документов, групповой обработки или реализации сложных бизнес-процессов, требующих явного разделения функций исполнителей. Тогда разные стадии отражения событий в учете реализуются не переходом по статусам у одного документа, а разными документами, которые вводятся на основании друг друга. В этой цепочке только определенные документы при проведении формируют движения.

Например, рассмотрим ситуацию, когда платежное поручение формируется в финансовом отделе, и при этом бухгалтер при проведении не должен изменять первичный документ. В этом случае, документ **Платежное поручение** не делает движений, а движения по платежному поручению формируются отдельным документом **Списание с расчетного счета**, который специально предназначен для автоматизированного формирования движений.

5. Непроведенные и помеченные на удаление документы не должны иметь [активных движений](https://its.1c.ua/db/content/v8std/src/100/300/i8100633.htm?_=1579516850).

6. Даже если документ не формирует движений, он должен проводиться, чтобы логически отличаться от «черновика».

7.1.1. В случае, если в документе не менялись данные, влияющие на проведение (например, изменили только значение реквизита **Комментарий**) проведение проведенного ранее документа, не должно приводить к изменению его движений.

Исключением из этого правила могут быть случаи, когда движения по регистру полностью или частично формируют внешние по отношению к документу алгоритмы (см. п.4).

7.1.2. При разработке алгоритмов формирования движений нужно стремиться избегать решений, когда результат формирования движений зависит от состояния учетных регистров, например, от остатков, т.к. в этом случае результат проведения будет зависеть от последовательности ввода документов.  
Исключением из этого правила могут быть отдельные, обоснованные случаи, когда сама суть алгоритма заключается в анализе последовательности, как, например, в алгоритмах реализующих партионный учет.

7.2. Для реализации поведения, описанного в п. 7.1, документы при формировании движений должны по максимуму опираться на данные, которые хранятся в этом документе. Данные, которые в документе не сохраняются, должны быть защищены от изменения. Это достигается реализацией следующего комплекса мер.

7.2.1. Если поддерживается изменение пользователем внешних, по отношению к документу, данных (например, реквизитов НСИ), влияющих на формирование движений, то значения этих реквизитов должны быть сохранены в документах.

В противном случае изменение этих данных должно быть заблокировано. В конфигурациях на основе **Библиотеки стандартных подсистем** для этого рекомендуется использовать возможности механизма **Запрет редактирования реквизитов**.

Исключения из этого правила описаны в п. 7.1.2.

7.2.2. Нужно стремиться, чтобы настройки программы (например, значения функциональных опций) оказывали наименьшее влияние на формирование движений. Тогда пользователь сможет свободно менять эти настройки.

Некоторые приемы для достижения такого поведения

* указание даты начала действия настройки (периода действия) и учет этой даты в алгоритмах формирования движений;
* заполнение отключенных по настройкам обязательных полей значениями по умолчанию: тогда пользователь сможет свободно включить настройку, ограничения будут связаны только с отключением такой настройки;
* формирование движений без учета настройки и дополнительные меры в объектах, которые отображают информацию из учетных регистров. Например, значение измерения регистра накопления всегда пишется одинаково, но отчеты по этому регистру измерение скрывают, если оно отключено.

7.2.3 Если все меры по исключению зависимости формирования движений от настроек программы исчерпаны, то необходимо предусмотреть одну из мер:

* автоматическая обработка данных, которая запускается в фоне после изменения пользователем настройки. О запуске такой обработки пользователя нужно предупреждать перед редактированием настройки;
* обработка данных, которая запускается пользователем вручную. Перед редактированием настройки пользователя нужно уведомлять о необходимости запуска обработки. Так же необходимо в интерфейсах, которые могут оперировать данными, подлежащими обработке (например, в отчетах) предупреждать пользователя, о необходимости запуска обработки;
* если предусмотреть обработку данных не представляется возможным, то при редактировании настройки необходимо предупреждать пользователя о том, что это делать не рекомендуется после начала ведения учета.

При этом допустимо поведение, когда реакция программы на включение и отключение настройки будет разной. Например, включение настройки проходит без предупреждения, а отключать ее не рекомендуется.

7.3. При изменении логики формирования движений для обеспечения выполнения условий п.7.1 необходимо предусмотреть [обработчики обновления информационной базы](https://its.1c.ua/db/content/v8std/src/800/i8100690.htm?_=1579516850) либо поддерживать для существующих на момент обновления документов старые алгоритмы формирования движений.

8. Для большинства событий отражение в учете может быть обратимым. В таком случае, для этого следует использовать механизм отмены проведения документов.

См. также

* [Имена объектов метаданных в конфигурациях](https://its.1c.ua/db/content/v8std/src/100/100/i8100550.htm?_=1579516850)
* [Порядок записи движений документов](https://its.1c.ua/db/content/v8std/src/300/300/i8100450.htm?_=1579516850)
* [Самодостаточность регистров](https://its.1c.ua/db/content/v8std/src/100/300/i8100477.htm?_=1579516850)

### Использование активности движений

#std633

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

В случае если в конфигурации применяются механизмы, использующие переключение активности движений определенных регистров (например, для ручной корректировки движений), то следует придерживаться следующего подхода.

1. Активность записей таких регистров должна учитываться во всех запросах и отчетах к ним. Для игнорирования неактивных записей условие выборки из регистра должно содержать дополнительное условие вида:

ГДЕ Активность = ИСТИНА

Если в запросе выбираются записи из виртуальных таблиц, связанных с регистрами, то это условие автоматически учитывается платформой **1С:Предприятие** (виртуальные таблицы регистров содержат только активные записи).

2. В частности, активность записей регистров должна учитываться в универсальных отчетах (или в любой универсальной бизнес-логике), которые поддерживают работу с произвольными регистрами конфигурации. Среди таких регистров могут оказаться и те, где используется переключение активности движений.

3. При отмене проведения документов, которые позволяют напрямую редактировать движения в своих регистрах (т.н. ручная корректировка движений), следует отключать активность движений, а не удалять их. Пример такого документа: «Операция (бухгалтерский и налоговый учет)», в котором имеется возможность «ручного» ввода операций.

См. также

* [Требования к проведению документов](https://its.1c.ua/db/content/v8std/src/100/300/i8100603.htm?_=1579516850)
* [Порядок записи движений документов](https://its.1c.ua/db/content/v8std/src/300/300/i8100450.htm?_=1579516850)

### Самодостаточность регистров

#std477

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

При разработке структуры регистров необходимо придерживаться правила, что регистр должен быть логически независим от регистраторов. Любая логика, опирающаяся или анализирующая данные регистра, а также любые отчеты по этому регистру не должны обращаться к полям регистратора, им должно быть достаточно данных самого регистра.

Обращение к полям регистратора "через точку" приводит к неявному соединению с дополнительными таблицами. Кроме того, в распределенной информационной базе регистратора может и не быть, если движения в регистрах мигрируют между узлами, а регистраторы - нет.

См. также

* [Разыменование ссылочных полей составного типа в языке запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100654.htm?_=1579516850)
* [Разработка планов обмена с отборами](https://its.1c.ua/db/content/v8std/src/700/i8100701.htm?_=1579516850)

### Реквизит «Комментарий» у документов

#std531 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Для всех документов рекомендуется создавать реквизит **Комментарий** (строка неограниченной длины). В этом реквизите пользователи могут записывать по документу различные заметки служебного характера, которые не относятся к прикладной специфике документа (например, причина пометки на удаления и т.п.). Доступ к реквизиту для пользователей должен быть настроен также как и к самому документу (если документ доступен только для чтения, то и комментарий – только для чтения; если же есть право записи документа, то и значение реквизита также можно изменять).

2. Если же штатный сценарий работы пользователя предусматривает внесение произвольной текстовой информации в документ, то для этого необходимо предусмотреть отдельные реквизиты «прикладного» характера. Например, в документе **Заказ клиента** для описания дополнительных договоренностей с клиентом следует предусмотреть реквизит **Дополнительная информация**, а не пользоваться служебным реквизитом **Комментарий**.

3. В простейшем случае, в качестве внешнего редактора текста комментария рекомендуется использовать функцию **ВвестиСтроку**. При использовании в конфигурации Библиотеки стандартных подсистем можно воспользоваться специализированной процедурой **ПоказатьФормуРедактированияКомментария** общего модуля **ОбщегоНазначенияКлиент**.

См. также

* [Поля "Ответственный" и "Комментарий" (8.2)](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/1%C2%A0400/i8100587.htm?_=1579516850)
* [Поля "Ответственный" и "Комментарий" (8.3)](https://its.1c.ua/db/content/v8std/src/1%C2%A0000/1%C2%A0300/i8100719.htm?_=1579516850)

### Удаление устаревших объектов метаданных из конфигурации

#std534

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Если при изменении структуры метаданных конфигурации планируется удалить объект метаданных (реквизит, измерение, ресурс и пр.), связанный с записями информационной базы, то необходимо принять решение об удалении или переносе данных этого объекта в новые структуры. При переносе данных в другие объекты рекомендуется придерживаться следующих правил.

1.1. Не удалять из конфигурации устаревшие объекты метаданных и реквизиты безвозвратно, а пометить их как устаревшие, добавив к их именам префикс **"Удалить"** (англ.**"Obsolete"**). Например: реквизит **"ОсновнойДоговор"**(англ. **"MainContract"**)  должен быть переименован в **"УдалитьОсновнойДоговор"**(англ. **"ObsoleteMainContract"**).

В синоним устаревшего объекта (реквизита) рекомендуется добавлять префикс **"(не используется)"**(англ. **"(not used)"**), например: **"(не используется) Основной договор"**(англ.**"(not used) Main contract"**). Если же устарел стандартный реквизит, то префикс **"(не используется)"** также добавляется в его синоним.

1.2. После изменения структуры метаданных следует обеспечить перенос данных из устаревших реквизитов в новую структуру метаданных конфигурации.

1.3. Если удаляемый объект метаданных является документом – регистратором движений, а соответствующие регистры с движениями остаются в составе конфигурации, то необходимо обратить внимание на необходимость сохранения движений. Для сохранения движений документов – устаревших объектов метаданных, рекомендуется:

* Запретить генерацию движений при проведении документов этого вида.
* Запретить снятие пометки удаления для документов этого вида.
* Во всех существующих движениях документов этого вида изменить регистратор на один или несколько замещающих документов-регистраторов: существующих универсальных или специально разработанных. Например **"Перенос данных",** **"Операция".**
* Пометить все документы этого вида на удаление.

1.4. Произвести замену во всей конфигурации обращений к устаревшим реквизитам на обращение к новым данным, поскольку использование устаревших объектов и их реквизитов после изменения структуры метаданных методически неверно. В частности, исключить устаревшие объекты метаданных из всех ролей ([кроме ролей **ПолныеПрава** и **АдминистраторСистемы**](https://its.1c.ua/db/content/v8std/src/600/i8100689.htm?_=1579516850)), подписок на события и т.п., а также удалить у них код, формы, макеты, команды и другие элементы, ставшие избыточными.

1.5. При сортировке устаревших объектов метаданных и реквизитов в дереве метаданных следует придерживаться общих требований к конфигурации.

1.6. Также рекомендуется выполнить очистку устаревших данных с тем, чтобы они не влияли на размер базы и не потребляли ресурсы (при резервном копировании, реструктуризации и других операциях).

В случае сложных (ошибкоемких) алгоритмов переноса данных, такую очистку целесообразно проводить не сразу, а через один или несколько релизов. Тем самым, остается возможность выпуска внепланового релиза для устранения последствий некорректной работы алгоритмов переноса.

2. Необходимость в переносе данных также может возникнуть при пересмотре структуры измерений регистров. Следует создать новый регистр с правильной структурой, а старый отметить как устаревший и перенести записи из старого регистра в новый в тех случаях, когда измерение регистра сведений становится не актуальным: удаляется, либо изменяется его тип, либо у измерения составного типа уменьшается состав типов.  
При этом создать новый регистр не требуется, если в регистр добавляется новое измерение или у измерения составного типа расширяется состав типов.

3. Безвозвратно удалять устаревшие объекты метаданных и реквизиты, помеченные префиксом **"Удалить"**(англ. **"Obsolete"**), следует при выпуске очередных версий конфигурации в том случае, если соблюдается одно из условий:

1. Переход со "старой" версии конфигурации на новые версии всегда выполняется пользователями последовательно, "через" версию с реализованным переносом данных из "устаревших" объектов метаданных и реквизитов. Например: если в конфигурации версии 1.1 реквизит **"ОсновнойДоговор"** был помечен как устаревший, то переход с версии 1.0 на версию 2.0 всегда выполняется только последовательно: сначала на версию 1.1 (в которой происходит обработка устаревших данных), а затем на 2.0 (в которой устаревшие данные могут быть удалены безвозвратно). Непосредственный переход с версии 1.0 на 2.0 технически невозможен (запрещен).
2. Вероятность того, что "старой" версией конфигурации еще пользуются, стала нулевой или пренебрежимо малой.

При этом может потребоваться выпустить промежуточный релиз, в котором обеспечить очистку устаревших данных - см. п.1.6. В противном случае, может завершиться ошибкой реструктуризация регистров, в измерениях которых остаются ссылки на устаревшие данные.

### Использование констант

#std632

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Следует выполнять запись константы вне транзакций, т.к. ее запись может стать «узким» местом при конкурентой работе.  
Например, неправильно:

// Увеличиваем счетчик проведенных документов в константе  
Процедура ОбработкаПроведения(Отказ, РежимПроведения)  
  ТекущееЗначение = Константы.СчетчикПроведенныхДокументов.Получить();  
  Константы.СчетчикПроведенныхДокументов.Установить(ТекущееЗначение + 1);  
КонецПроцедуры

На время записи значения в константу, работа других сеансов приостанавливается, если в это же время они выполняют запись этой же константы. Подробнее о причинах избыточных блокировок и методах оптимизации см. базу знаний [«Технологические вопросы крупных внедрений»](http://kb.1c.ru/).

Вместе с тем, недопустимо решать проблему блокировок констант другими методами, в частности, тотальным кешированием констант в [параметрах сеанса](https://its.1c.ua/db/content/v8std/src/100/100/i8100413.htm?_=1579516850) или в функциях [общих модулей с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100724.htm?_=1579516850).

### Работа с неактуальными (недействительными) объектами

#std638 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Данная рекомендация действует для ситуаций, когда какой-либо объект информационной базы перестает быть актуальным – навсегда или на время (сотрудник увольняется или уходит в декретный отпуск, подразделение расформируется и т.д.). При этом удалять этот объект из базы недопустимо, чтобы не нарушать ссылочную целостность: на него должны ссылаться другие ранее созданные объекты. Например, объект типа **Файл** должен содержать в поле **Автор** ссылку на уволившегося сотрудника и после его увольнения.

2. Для запрета выбора неактуальных объектов с помощью автоподбора и быстрого выбора в полях ввода следует выбрать один из двух подходов к реализации (2.1 или 2.2). Проиллюстрируем их далее на примере пользователей информационной системы, учет которых ведется в справочнике **Пользователи**.

Для учета неактуальных пользователей у справочника **Пользователи** добавлен реквизит **Недействителен** (**Булево**), по умолчанию **Ложь**.

2.1. Если запрет должен действовать во всех или в большинстве полей ввода форм системы, то он задается по умолчанию.

2.1.1. В модуле менеджера справочника **Пользователи** реализуются обработчики **ОбработкаПолученияДанныхВыбора** и **ОбработкаПолученияФормы** для установки параметров отбора. Пример реализации этих обработчиков для справочника **Пользователи**:

Процедура ОбработкаПолученияДанныхВыбора(ДанныеВыбора, Параметры, СтандартнаяОбработка)  
 Если Не Параметры.Отбор.Свойство("Недействителен") Тогда  
  Параметры.Отбор.Вставить("Недействителен", Ложь);  
 КонецЕсли;  
КонецПроцедуры

Процедура ОбработкаПолученияФормы(ВидФормы, Параметры, ВыбраннаяФорма, ДополнительнаяИнформация, СтандартнаяОбработка)  
 Если ВидФормы = "ФормаВыбора" Тогда   
  ПараметрИзменен = Ложь;    
  Если Не Параметры.Свойство("Отбор") Тогда  
   Параметры.Вставить("Отбор", Новый Структура("Недействителен", Ложь));  
   ПараметрИзменен = Истина;  
  ИначеЕсли Не Параметры.Отбор.Свойство("Недействителен") Тогда  
   Параметры.Отбор.Вставить("Недействителен", Ложь);  
   ПараметрИзменен = Истина;  
  КонецЕсли;  
    
  // Этот код нужен, чтобы были использованы измененные нами значения параметров  
  Если ПараметрИзменен Тогда    
   СтандартнаяОбработка = Ложь;  
   ВыбраннаяФорма = "ФормаВыбора"; // передаем имя формы выбора  
  КонецЕсли;  
 КонецЕсли;  
КонецПроцедуры

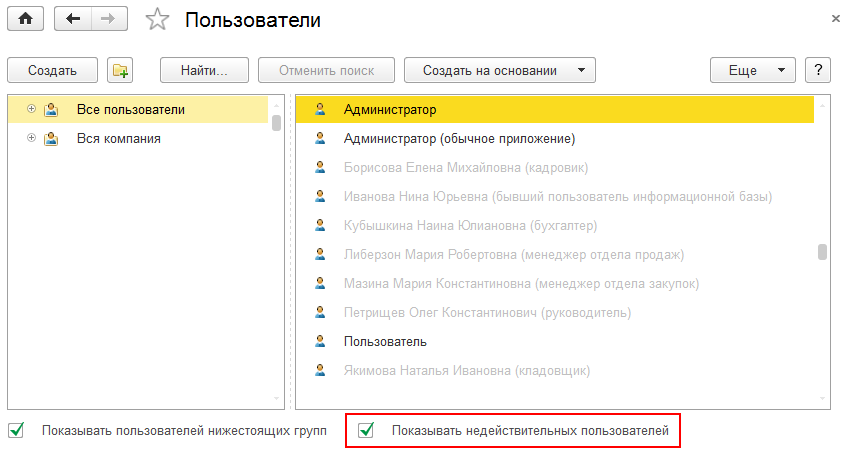
2.1.2. Для тех реквизитов, где это поведение нужно изменить (например, нужно выводить всех пользователей или должно работать другое ограничение) следует явно установить свойства **«Параметры выбора»** и **«Связи параметров выбора»** с необходимыми в конкретном контексте значениями выбора:

* если требуется выбирать недействующих пользователей, указываем в свойстве реквизита **«Параметры выбора»** значение **Отбор.Недействителен(Ложь)**,
* а если не требуется ограничивать выбор вообще, то оба значения - **Истина** и **Ложь**.

2.2. Если запрет на выбор неактуальных объектов сильно зависит от контекста (сценариев работы), то не следует его устанавливать по умолчанию.

* Модуль менеджера справочника **Пользователи**не реализуется.
* В простейшем случае, во всех объектах, в которых есть реквизиты типа **СправочникСсылка.Пользователи** устанавливаются значения свойств **«Параметры выбора»** и **«Связи параметров выбора»**, как описано выше в пункте 2.1.2.
* В тех случаях, когда критерий ограничения не может быть описан параметрами выбора, то реализуются обработчики формы **ОбработкаПолученияДанныхВыбора**, **ОбработкаВыбора** и **ОкончаниеВводаТекста**, а также разрабатывается отдельная форма выбора, в которой реализуется та же логика ограничения.

3. В формах списка и выбора пользователей рекомендуется добавить флажок «Показывать недействительных пользователей». С его помощью возможно выбрать или открыть карточку пользователя, а также снова сделать пользователя действительным (например сотрудницу, вернувшуюся из декретного отпуска).

4. Для отображения неактуальных объектов в списках рекомендуется использовать элемент стиля **ТекстЗапрещеннойЯчейкиЦвет** (192,192,192).

### Использование предопределенных элементов

#std697

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Действует для версии платформы 1С:Предприятие 8.3.3 и выше без режима совместимости с версией 8.2

1.1. В справочниках, планах счетов, планах видов характеристик и планах видов расчета имеется возможность создавать предопределенные элементы автоматически или программно.

1.2. В большинстве случаев, предопределенные элементы рекомендуется создавать автоматически, поскольку они постоянно нужны и требуется упростить обращение к этим элементам из кода.  
Например, предопределенная страна **Россия** в справочнике **Страны мира**, предопределенные профиль групп доступа **Администратор** и т.п.

Для этого

* в свойстве **ОбновлениеПредопределенныхДанных** справочника, плана счетов, плана видов характеристик или плана видов расчета должно быть установлено значение **Авто** (по умолчанию), а также не следует допускать программных вызовов метода **УстановитьОбновлениеПредопределенныхДанных** этих объектов для переключения этого режима.
* запретить удаление предопределенных элементов пользователями, выключив во всех ролях [следующие права](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1579516850) (по умолчанию выключены):
  + **ИнтерактивноеУдалениеПредопределенныхДанных**
  + **ИнтерактивнаяПометкаУдаленияПредопределенныхДанных**
  + **ИнтерактивноеСнятиеПометкиУдаленияПредопределенныхДанных**
  + **ИнтерактивноеУдалениеПомеченныхПредопределенныхДанных**

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1.3. Исключение составляют дочерние узлы РИБ, в котором предопределенные элементы автоматически не создаются (и не обновляются при изменении в метаданных), а должны быть переданы из главного узла вместе с изменениями конфигурации.  При этом:  а) конфигурация должна обеспечивать загрузку сообщения обмена в подчиненный узел РИБ до выполнения другого прикладного кода, который обращается к получаемым из главного узла предопределенным элементам;  б) в прикладной логике загрузки данных из главного узла (обработчик события **ПриПолученииДанныхОтГлавного**, правила регистрации объектов) следует избегать обращений к предопределенным элементам, поскольку нет гарантии, что они уже были загружены из сообщения обмена;  в) код обработчиков обновления ИБ, который обрабатывает предопределенные элементы, не должен выполняться в подчиненных узлах РИБ:  Если ПланыОбмена.ГлавныйУзел() = Неопределено Тогда  // заполняем предопределенные элементы  // ... КонецЕсли;  При использовании в конфигурации подсистемы "Обмен данными" Библиотеки стандартных подсистем (БСП) версии 2.1.4 и выше требования (а) и (б) снимаются. |

1.4. Для таблиц с предопределенными элементами, которые не входят в состав плана обмена РИБ (и на которые не ссылаются другие таблицы, входящие в состав плана обмена РИБ) свойство **ОбновлениеПредопределенныхДанных** необходимо устанавливать в значение **ОбновлятьАвтоматически**.

Значение **Авто** в этом случае не подходит, так как для подчиненного узла **Авто** означает **НеОбновлятьАвтоматически**, то есть предопределенные элементы таблицы автоматически созданы не будут.

При включенном режиме совместимости с версией 8.3.3 также необходимо при первом запуске подчиненного узла РИБ (сразу после того, как была обновлена его конфигурация) устанавливать автоматическое обновление в данных с помощью вызова:

Справочники.<ИмяСправочника>.УстановитьОбновлениеПредопределенныхДанных(ОбновлениеПредопределенныхДанных.ОбновлятьАвтоматически);

2. В некоторых случаях, предопределенные элементы не требуется создавать автоматически, если их наличие зависит от какого-либо условия: включенной функциональной опции, режима работы программы и т.п.

Например, те или иные предопределенные виды расчетов в плане видов расчета **Начисления** зависят от значений функциональных опций **ИспользоватьУчетВремениСотрудниковВЧасах**, **ИспользоватьСдельныйЗаработок** и др.

Для этого

* в свойстве **ОбновлениеПредопределенныхДанных** справочника, плана счетов, плана видов характеристик или плана видов расчета нужно установить в значение "Не обновлять автоматически"
* предусмотреть код создания (и [пометки недействительным](https://its.1c.ua/db/content/v8std/src/100/300/i8100638.htm?_=1579516850)) предопределенного элемента в зависимости от бизнес-логики, например:

Если ПолучитьФункциональнуюОпцию("ИспользоватьУчетВремениСотрудниковВЧасах") Тогда  
 НачислениеОбъект = ПланыВидовРасчета.Начисления.СоздатьВидРасчета();  
 НачислениеОбъект.ИмяПредопределенныхДанных = "ОкладПоЧасам";  
 // ...  
 НачислениеОбъект.Записать();  
КонецЕсли;

* учитывать в прикладном коде отсутствие предопределенных элементов в ИБ. В противном случае, при обращении к несуществующему предопределенному элементу из кода или текста запроса будет вызвано исключение:

  ... = ПланВидовРасчета.Начисления.ОкладПоЧасам;  
  ... = ПредопределенноеЗначение("ПланВидовРасчета.Начисления.ОкладПоЧасам");

При использовании в конфигурации Библиотеки стандартных подсистем (БСП) версии 2.1.4 и выше рекомендуется использовать функцию **ПредопределенныйЭлемент** общего модуля **ОбщегоНазначения** или **ОбщегоНазначенияКлиент**, которая возвращает **Неопределено** для несуществующих в ИБ предопределенных элементов:

  ... = ОбщегоНазначенияКлиент.ПредопределенныйЭлемент("ПланВидовРасчета.Начисления.ОкладПоЧасам");

См. также

* [Получение предопределенных значений на клиенте](https://its.1c.ua/db/content/v8std/src/400/i8100443.htm?_=1579516850)

## [Обработчики событий объектов](https://its.1c.ua/db/v8std/browse/13/-1/1/5)

### Обработчик события ПередЗаписью

#std464

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В данном обработчике модуля объекта, как правило, должны выполняться действия, связанные с заполнением значений реквизитов объекта (набора записей, значения константы; далее упрощенно - "объект"), проверки правильности их заполнения, связанности состояния объекта с некоторыми внешними данными. Также в данном обработчике следует выполнять действия, связанные с обращением к "старым" значениям реквизитов объекта, сохраненным в базу данных (имеет смысл при редактировании уже записанных ранее объектов, наборов записей и т.п.).

См. также [раздел «Проверки, выполняемые в и вне транзакции записи объекта» статьи «Обработчик события ОбработкаПроверкиЗаполнения»](https://its.1c.ua/db/content/v8std/src/100/400/i8100463.htm?_=1579516850)

2. Все действия в процедуре-обработчике события**ПередЗаписью** должны выполняться после [проверки на **ОбменДанными.Загрузка**](https://its.1c.ua/db/content/v8std/src/100/400/i8100773.htm?_=1579516850).

### Обработчик события ПриЗаписи

#std465

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В данном обработчике модуля объекта (набора записей, значения константы; далее - "объект"), как правило, выполняются действия по записи связанной с объектом данных в других объектах конфигурации, а также выполняются другие действия, связанные с изменением объекта.  
  
Запрещается в данном обработчике изменять содержимое записываемого объекта, поскольку на момент выполнения обработчика, объект уже записан в БД.

2. Все действия в процедуре-обработчике события **ПриЗаписи** должны выполняться после [проверки на **ОбменДанными.Загрузка**](https://its.1c.ua/db/content/v8std/src/100/400/i8100773.htm?_=1579516850).

См. также

* [Раздел «Проверки, выполняемые в и вне транзакции записи объекта» статьи «Обработчик события ОбработкаПроверкиЗаполнения»](https://its.1c.ua/db/content/v8std/src/100/400/i8100463.htm?_=1579516850)

### Обработчик события ПередУдалением

#std752

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В данном обработчике модуля объекта, как правило, должны выполняться действия, которые необходимо выполнить перед удалением объекта. Например, перед удалением присоединенного файла может потребоваться произвести очистку ссылок на этот файл в объекте-владельце.

2. Все действия в процедуре-обработчике события **ПередУдалением** должны выполняться после [проверки на **ОбменДанными.Загрузка**](https://its.1c.ua/db/content/v8std/src/100/400/i8100773.htm?_=1579516850).

Т. е. они не должны выполняться перед удалением объекта через механизм обмена данными, так как это может привести к ошибкам. Примером таких ошибок является обращение к предопределенным объектам после очистки области данных.

### Обработчик события ПриКопировании

#std466

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

В данном обработчике модуля объекта выполняются действия, связанные с созданием нового объекта путем копирования.

В частности, в данном обработчике выполняются действия по очистке содержимого реквизитов объекта в случаях, когда значения этих реквизитов не должны сохраняться при копировании.

### Обработчик события ОбработкаПроверкиЗаполнения

#std463

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. В данном обработчике модуля объекта выполняются действия, связанные с проверкой правильности заполнения значений реквизитов объектов (измерений, ресурсов, реквизитов табличных частей и т.п., далее: просто "реквизиты").

См. также [Подсказка и проверка заполнения](https://its.1c.ua/db/content/v8std/src/100/300/i8100478.htm?_=1579516850)

1.2. Данным обработчиком следует пользоваться в случаях, когда для проверки корректности значений реквизитов обычной проверки на заполненность уже недостаточно (например, значение реквизита логически связано со значением другого реквизита), или же требование к тому, чтобы значение реквизита было заполнено не является безусловным.

Если проверка заполнения какого-либо реквизита - условная (т.е. зависит от значений других реквизитов или значения параметризированной функциональной опции) в обработчике следует предусмотреть код, который удаляет имя такого реквизита из массива проверяемых реквизитов **ПроверяемыеРеквизиты**. В общем виде, схема проверки заполнения выглядит следующим образом:

* создать массив**НепроверяемыеРеквизиты**;
* в процессе проверки условий, добавлять в этот массив имена непроверяемых реквизитов (табличных частей);
* вызвать процедуру для удаления непроверяемых реквизитов (текст процедуры **УдалитьНепроверяемыеРеквизитыИзМассива** приведен ниже).

При этом не рекомендуется использовать другие схемы проверки заполнения значений реквизитов, так как они затрудняют анализ логики работы конфигурации, поскольку скрывают из свойства "Проверка заполнения" случаи условной проверки заполнения значений объектов.  
Например, неправильно:

Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)  
  
  ...  
  
  // Проверка значения реквизита на соответствие некоторым требованиям  
  Если НЕ ИННСоответствуетТребованиям(ИНН) Тогда  
    Сообщение = Новый СообщениеПользователю();  
    Сообщение.Текст = НСтр("ru = 'ИНН задан неверно.'");  
    Сообщение.Поле = "ИНН";  
    Сообщение.УстановитьДанные(ЭтотОбъект);  
    Сообщение.Сообщить();  
    Отказ = Истина;  
  КонецЕсли;  
  
  ...

  // Значение реквизита не должно быть пустым в зависимости от значения другого реквизита  
  Если ЮрФизЛицо = Перечисления.ЮрФизЛицо.ФизЛицо Тогда  
    // Для индивидуального предпринимателя должно быть сопоставлено физ. лицо  
    ПроверяемыеРеквизиты.Добавить("ИндивидуальныйПредприниматель");  
  КонецЕсли;  
  
  ...  
  
КонецПроцедуры

правильно:

Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)   
  
  НепроверяемыеРеквизиты = Новый Массив();  
  ...

  // Проверка значения реквизита на соответствие некоторым требованиям  
  Если НЕ ИННСоответствуетТребованиям(ИНН) Тогда  
    Сообщение = Новый СообщениеПользователю();  
    Сообщение.Текст = НСтр("ru = 'ИНН задан неверно.'");  
    Сообщение.Поле = "ИНН";  
    Сообщение.УстановитьДанные(ЭтотОбъект);  
    Сообщение.Сообщить();  
    Отказ = Истина;   
    НепроверяемыеРеквизиты.Добавить("ИНН");   
  КонецЕсли;

  ...  
  // Значение реквизита не должно быть пустым в зависимости от другого реквизита  
  Если ЮрФизЛицо <> Перечисления.ЮрФизЛицо.ФизЛицо Тогда    
    НепроверяемыеРеквизиты.Добавить("ИндивидуальныйПредприниматель");  
  КонецЕсли;

  ...  
  УдалитьНепроверяемыеРеквизитыИзМассива(ПроверяемыеРеквизиты, НепроверяемыеРеквизиты);  
КонецПроцедуры;

Процедура УдалитьНепроверяемыеРеквизитыИзМассива(МассивРеквизитов, МассивНепроверяемыхРеквизитов) Экспорт

  Для Каждого ЭлементМассива Из МассивНепроверяемыхРеквизитов Цикл  
   
    // перед удалением реквизита из массива необходимо проверить, что он там есть  
    // (не был удален ранее платформой или в коде).  
    ПорядковыйНомер = МассивРеквизитов.Найти(ЭлементМассива);  
    Если ПорядковыйНомер <> Неопределено Тогда  
      МассивРеквизитов.Удалить(ПорядковыйНомер);  
    КонецЕсли;  
   
  КонецЦикла;  
   
КонецПроцедуры

1.3. Следует учитывать, что обработчик **ОбработкаПроверкиЗаполнения** вызывается не при каждой записи объекта, в частности, он не вызывается в случаях если запись были инициирована программно.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1.4. В случае использования в конфигурации подсистемы "Обмен данными" Библиотеки стандартных подсистем обработчик **ОбработкаПроверкиЗаполнения** вызывается при проведении документов, после их загрузки из сообщения обмена. Для отключения некоторых проверок в этом режиме в обработчике можно анализировать дополнительное свойство объекта **ДополнительныеСвойства**.**ОтложенноеПроведение**. |

#### Проверки, выполняемые в и вне транзакции записи объекта

2.1. Проверки в обработчике **ОбработкаПроверкиЗаполнения** выполняются вне транзакции записи объекта. Поскольку в случае некорректного заполнения объекта выполнение операции будет прервано еще до записи объекта в базу данных, то размещение проверок в этом обработчике является наиболее эффективным.

При выполнении внетранзакционных проверок в обработчике **ОбработкаПроверкиЗаполнения** необходимо учитывать тот факт, что новое состояние объекта еще не записано. Если требуется выполнить запрос к тем или иным данным системы, например, прочитать признак **ВидНоменклатуры** для товаров, выбранных в табличной части документа, "отталкиваясь" от данных документа, то такую поверку можно выполнить, применяя сохранение необходимых для запроса данных во временные таблицы.

2.2. В то же время, в обработчике **ОбработкаПроверкиЗаполнения** не следует размещать проверки, которые должны гарантировать целостное состояние объекта или зависящих от него данных (например, движений) на которые рассчитывает система. Поэтому для реквизитов, некорректные значения которых могут привести к рассогласованности данных в информационной базе, проверку корректности следует выполнять в обработчиках событий, возникающих в транзакции записи - [**ПередЗаписью**](https://its.1c.ua/db/content/v8std/src/100/400/i8100464.htm?_=1579516850), [**ПриЗаписи**](https://its.1c.ua/db/content/v8std/src/100/400/i8100465.htm?_=1579516850), **ОбработкаПроведения** (для документов).

Для транзакционных проверок, в свою очередь, выделяются два случая:

1. Проверка состояния движений, формируемых документами оперативного учета. Такие проверки довольно часто встречаются в приложениях с оперативным учетом.
2. Проверка состояния других объектов информационной базы, ссылки на которых содержатся в текущем объекте. Такие проверки следует применять очень редко. Не следует злоупотреблять количеством проверок в транзакции записи объекта. Следует помнить, что внутри транзакции записи имеет смысл выполнять только проверки таких ресурсов или таких правил соответствия объектов друг другу, которые не изменяются без проверок всеми участниками процесса.

В первом случае, проверку остатков некоторого ресурса имеет смысл выполнять в транзакции записи только в том случае, если все документы выполняют такую же проверку в транзакции записи. Если хоть один из документов, изменяющих ресурс, делает это без проверок, выполнение проверок другими участниками процесса бессмысленно и такие проверки необходимо выполнять вне транзакции. Исключением может быть только случай, когда документ, который выполняет изменение контролируемого ресурса без проверок, вводится крайне редко. Например, не смотря на то, что документ "Инвентаризация товаров" изменяет остатки товаров без проверок, эта ситуация допустима в виду того, что он вводится крайне редко. Каждое такое исключение из правила должно быть оправданным.

Во втором случае, если при записи **Подразделения** в транзакции записи выполняется проверка, что сотрудник, выбранный в качестве руководителя подразделения, имеет должность "Руководитель", то при записи **Сотрудника** также должна выполняться и "встречная" проверка этого же правила: нельзя записать **Сотрудника** с должностью отличной от "Руководитель", если он указан руководителем того или иного подразделения. Поскольку правило, что "Сотрудник", выбранный руководителем подразделения, должен иметь должность "Руководитель", может быть нарушено как при записи подразделения, так и при записи сотрудника, то и проверка должна выполняться или в транзакции записи обоих объектов, или вне транзакции записи обоих объектов (а может и не выполняться вообще).

### Обработчик события ОбработкаЗаполнения

#std396

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В случае если в силу каких-либо условий необходимо ограничивать ввод на основании по команде "Создать на основании", то такую проверку следует выполнять в обработчике **ОбработкаЗаполнения**модуля объекта (набора записей). Например, это могут быть проверки вида:

* Для команды "Создать на основании" не различимы группы и элементы справочников и планов видов характеристик: команда одинаково доступна в форме списка как для групп, так и для элементов. Требуется запретить ввод на основании групп.
* Требуется запретить ввод на основании непроведенных документов.

2. Для оповещения пользователя о причинах отказа, в обработчике **ОбработкаЗаполнения** следует использовать исключения:

Процедура ОбработкаЗаполнения(ДанныеЗаполнения, ТекстЗаполнения, СтандартнаяОбработка)   
  
   Если ТипЗнч(ДанныеЗаполнения) = Тип("СправочникСсылка.Сотрудники") Тогда  
      Если ПолучитьЗначениеРеквизита(ДанныеЗаполнения, "ЭтоГруппа") = Истина Тогда  
         ВызватьИсключение "Ввод приказа о приеме на основании группы сотрудников невозможен!  
                           |Выберите сотрудника. Для раскрытия группы используйте клавиши Ctrl и стрелку вниз";   
      КонецЕсли;   
  
      // обработка заполнения объекта по данным заполнения   
  
   КонецЕсли;  
  
КонецПроцедуры

При этом не рекомендуются какие-либо иные решения для подобных проверок. В частности, не следует создавать дополнительные команды для ввода на основании и размещать проверки в обработчиках этих команд.

|  |
| --- |
| Методическая рекомендация (полезный совет)  3. Рекомендуется придерживаться следующей логической структуры обработчика **ОбработкаЗаполнения** (отдельные шаги могут быть пропущены):  3.1. Выполнение специального заполнения в зависимости от типа параметра **ДанныеЗаполнения**.  Например:  ТипДанныхЗаполнения = ТипЗнч(ДанныеЗаполнения); Если ТипДанныхЗаполнения = Тип("Структура") Тогда    ЗаполнитьДокументПоОтбору(ДанныеЗаполнения); ИначеЕсли ТипДанныхЗаполнения = Тип("ДокументСсылка.ЗаказКлиента") Тогда    ЗаполнитьДокументНаОснованииЗаказаКлиента(ДанныеЗаполнения); // ...  3.2. Выполнение общего заполнения, с целью заполнить значениями по умолчанию реквизиты, которые не были заполнены специальным заполнением. При этом необходимо предварительно проверять реквизит на заполненность.  Например:  Если Не ЗначениеЗаполнено(Подразделение) Тогда    Подразделение = ЗначениеНастроекПовтИсп.ПодразделениеПоУмолчанию(); КонецЕсли;  Также при заполнении реквизитов значениями по умолчанию следует, по возможности, использовать свойство метаданных **"Значение заполнения"**. Значение, указанное в этом свойстве будет автоматически присваиваться реквизиту при выходе из обработчика **ОбработкаЗаполнения**, в случае если параметр **СтандартнаяОбработка** установлен в **Истина**, и реквизит не был заполнен в обработчике.  Выполнение данной рекомендации позволит уменьшить количество логических ошибок заполнения и повысит читаемость кода. |

* [Перехват исключений в коде](https://its.1c.ua/db/content/v8std/src/300/200/i8100499.htm?_=1579516850)
* [Обращение из кода к автоматически формируемым элементам управления формы](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100536.htm?_=1579516850)

### Обработчики событий ОбработкаПолученияПредставления и ОбработкаПолученияПолейПредставления

#std746

Область применения: управляемое приложение, мобильное приложение.

1. С помощью данных обработчиков модуля менеджера объекта можно переопределить представление объекта информационной базы, которое выводится в полях форм и в списках.  
Пример реализации:

Процедура ОбработкаПолученияПолейПредставления(Поля, СтандартнаяОбработка)  
 Поля.Добавить("Наименование");  
 Поля.Добавить("Дата");  
 СтандартнаяОбработка = Ложь;  
КонецПроцедуры

Процедура ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка)  
 Наименование = ?(ПустаяСтрока(Данные.Наименование), НСтр("ru = 'Без описания'"), Данные.Наименование);  
 Дата = Формат(Данные.Дата, ?(ПолучитьФункциональнуюОпцию("ИспользоватьДатуИВремяВСрокахЗадач"), "ДЛФ=DT", "ДЛФ=D"));  
 Представление = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru = '%1 от %2'"), Наименование, Дата);  
 СтандартнаяОбработка = Ложь;  
КонецПроцедуры

2. Обработчики вызываются при любой необходимости получения представления какого-либо объекта информационной базы. Поэтому избыточные данные или неправильный выбор данных для формирования представления могут привести к существенному замедлению работы системы.

Также не следует выполнять в этих обработчиках запросы или получение объектов информационной базы (в том числе, запрещены обращения к реквизитам объектов ссылочных типов через точку, что приводит к чтению всего объекта целиком из базы данных). Также нежелательно использовать получение представления и реквизитов ссылок.

3. Обработчики могут быть также вызваны, в том числе, при записи и удалении объекта в режиме обмена данными. Представление объекта при этом запрашивается для формирования записи в журнал регистрации. Поэтому к ним предъявляются те же требования, что и к логике регистрации объектов  - см. п. 1.2 [Разработка планов обмена с отборами](https://its.1c.ua/db/content/v8std/src/700/i8100701.htm?_=1579516850).

Эти требования также справедливы при разработке планов обмена для синхронизации с другими программами (не РИБ, по правилам конвертации) с помощью подсистемы «Обмен данными» Библиотеки стандартных подсистем.  
Например, недопустимо [обращаться к предопределенным элементам](https://its.1c.ua/db/content/v8std/src/100/300/i8100697.htm?_=1579516850), которые еще могли быть не загружены в базу или, наоборот, уже удалены в ходе обмена данными:

Процедура ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка)

    СтандартнаяОбработка = Ложь;  
    Если Данные.ВидОбразования = Справочники.ВидыОбразованияФизическихЛиц.ПослевузовскоеОбразование Тогда  
        Представление = НСтр("ru = 'Послевузовское образование'");  
    Иначе  
      ...

4. При реализации обработчиков следует также учитывать требования о поддержке [толстого клиента, управляемое приложение, клиент-сервер](https://its.1c.ua/db/content/v8std/src/100/100/i8100680.htm?_=1579516850).

См. также

* [Получение представлений для ссылочных значений в табличном документе](https://its.1c.ua/db/content/v8std/src/300/300/i8100449.htm?_=1579516850)
* [Пользовательские представления объектов](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/i8100468.htm?_=1579516850)

### Использование признака ОбменДанными.Загрузка в обработчиках событий объекта

#std773

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Все действия в процедурах-обработчиков событий [ПередЗаписью](https://its.1c.ua/db/content/v8std/src/100/400/i8100464.htm?_=1579516850), [ПриЗаписи](https://its.1c.ua/db/content/v8std/src/100/400/i8100465.htm?_=1579516850), [ПередУдалением](https://its.1c.ua/db/content/v8std/src/100/400/i8100752.htm?_=1579516850) должны выполняться после проверки на **ОбменДанными.Загрузка**:

Процедура ПередЗаписью(Отказ)  
Если ОбменДанными.Загрузка Тогда  
     Возврат;  
КонецЕсли;

// код обработчика  
// ...  
КонецПроцедуры

Это необходимо для того, чтобы никакая бизнес-логика объекта не выполнялась при записи объекта через механизм обмена данными, поскольку она уже была выполнена для объекта в том узле, где он был создан. В этом случае все данные загружаются в ИБ «как есть», без искажений (изменений), проверок или каких-либо других дополнительных действий, препятствующих загрузке данных.

Кроме механизма обмена данными есть и другие случаи, когда это должно быть так. В общем виде следует руководствоваться следующим подходом: механизмы, не рассчитанные на особенности конкретных конфигураций, должны иметь возможность загрузить данные при установленном флаге загрузки так, как будто текста обработчика нет вообще:

Объект.ОбменДанными.Загрузка = Истина;  
Объект.Записать();

Например, требуется загрузить всю базу из XML «как есть». Для этого должно быть достаточно установить записываемым объектам **ОбменДанными.Загрузка = Истина** и все данные должны загрузиться без искажений, проверок и дополнительных  действий, т. е. так же как и при пустом обработчике.

2. Исключение составляет механизм обмена данными, который в ходе загрузке данных в базу регистрирует эти данные к выгрузке на других узлах плана обмена.

В тех случаях, когда в конфигурации используется подсистема «Обмен данными» БСП, и возникла необходимость отключить ее, следует устанавливать дополнительное свойство **ОтключитьМеханизмРегистрацииОбъектов**:

Объект.ОбменДанными.Загрузка = Истина;  
Объект.ДополнительныеСвойства.Вставить("ОтключитьМеханизмРегистрацииОбъектов");  
Объект.Записать();

В случае других исключений, причина исключения из этого правила должна быть описана в комментарии к выполняемым действиям.

3. Требования выше также распространяются на обработчики подписок на эти события.

4. При этом вызывающая сторона, выставляя записываемому объекту признак **ОбменДанными.Загрузка** в **Истина**, берет на себя ответственность за целостность данных этого объекта.

Например, при записи объекта через механизм обмена данными в РИБ это обеспечивается корректным состоянием объекта в том узле, где он был создан (или изменен).

В других случаях вызывающая сторона должна принять меры по корректному заполнению записываемого объекта. Например, при загрузке данных через механизм обмена данными по правилам конвертации или с помощью формата [EnterpriseData](https://its.1c.ua/db/content/v8std/src/700/i8100771.htm?_=1579516850), следует выполнять все необходимые действия по (до)заполнению объекта. Эти действия рекомендуется размещать в экспортных процедурах самого объекта, которые используются вызывающей стороной при записи объекта в режиме обмена данными.

## [Регламентные задания](https://its.1c.ua/db/v8std/browse/13/-1/1/6)

### Общие требования к регламентным заданиям

#std540

Область применения: управляемое приложение, обычное приложение.

1. В общем случае, регламентные задания следует использовать, когда необходимо выполнить определенные периодические или однократные действия в соответствии с расписанием.

2. При этом если регламентные задания не требуется добавлять или удалять в зависимости от действий пользователя или логики конфигурации, следует использовать предопределенные регламентные задания. Такие задания автоматически создаются в информационной базе с тем расписанием и состоянием, которое было задано разработчиком в Конфигураторе. Примеры предопределенных регламентных заданий:

* загрузка курсов валют;
* извлечение текста для полнотекстового индексирования;
* обновление агрегатов.

3.1. Если выполнение регламентного задания зависит от включенных одной или нескольких функциональных опций (ФО), то необходимо программно управлять признаком предопределенного регламентного задания Использование в зависимости от установленных ФО. Иначе регламентное задание будет приводить к запуску сеанса, занимая вычислительные ресурсы сервера 1С:Предприятие.

Например, имеем регламентное задание **ПолучениеИОтправкаЭлектронныхПисем** (с установленным флажком **Использование**), которое должно выполняться только в том случае, если установлена ФО **ИспользоватьПочтовыйКлиент**.

Неправильно: создавать предопределенное регламентное задание, зависящее от ФО, с установленным флажком **Использование**.

Правильно: снять флажок **Использование** и управлять использованием регламентного задания в зависимости от включения/выключения функциональной опции.  
Если в конфигурации используется подсистема «Регламентные задания» Библиотеки стандартых подсистем (БСП), то для такой настройки следует использовать процедуру **ПриОпределенииНастроекРегламентныхЗаданий** общего модуля **РегламентныеЗаданияПереопределяемый**. Например:

Настройка = Настройки.Добавить();  
Настройка.РегламентноеЗадание = Метаданные.РегламентныеЗадания.ОбновлениеСтатусовДоставкиSMS;  
Настройка.ФункциональнаяОпция = Метаданные.ФункциональныеОпции.ИспользоватьПочтовыйКлиент;  
Настройка.ДоступноВМоделиСервиса = Ложь;

После чего в состав определяемого типа **МестоХраненияФункциональныхОпций** необходимо добавить константы, соответствующие функциональным опциям, используемым для управления регламентными заданиями.

Для конфигураций без БСП следует управлять использованием регламентного задания, разместив, например, в модуле менеджера значения константы **ИспользоватьПочтовыйКлиент** следующий код:

Процедура ПриЗаписи(Отказ)  
   
 Задание = РегламентныеЗадания.НайтиПредопределенное(Метаданные.РегламентныеЗадания.ПолучениеИОтправкаЭлектронныхПисем);  
   
 Если Задание.Использование <> Значение Тогда  
  Задание.Использование = Значение;  
  Задание.Записать();  
 КонецЕсли;  
   
КонецПроцедуры

3.2. Дополнительно следует обезопасить выполнение регламентного задания, включенного через консоль или другим способом, минуя включение ФО, вставив в начало процедуры обработки регламентного задания следующий код:

ОбщегоНазначения.ПриНачалеВыполненияРегламентногоЗадания();  
Если НЕ ПолучитьФункциональнуюОпцию("ИспользоватьПочтовыйКлиент") Тогда  
 ВызватьИсключение НСтр("ru = 'Регламентное задание недоступно по функциональным опциям.'");  
КонецЕсли;

Если в конфигурации используется подсистема «Регламентные задания» БСП и настроены зависимости регламентных заданий от ФО (как указано в п.3.1), то вместо этого достаточно вставить вызов, как показано в п.6.

4.1. Если выполнение регламентного задания зависит от данных информационной базы, то флажок **Предопределенное** у регламентного задания следует отключать.  
Например:

* обмен данными с другими информационными базами должен проводиться с каждой базой по индивидуальному расписанию;
* запуск каждой дополнительной обработки в базе требуется выполнять по отдельному расписанию.

В этих случаях требуется создавать экземпляры регламентных заданий и параметризовать их объектами ИБ (например, узлами ИБ, элементами справочника **Дополнительные обработки** и т.п.) из кода на встроенном языке с помощью метода **РегламентныеЗадания**.**СоздатьРегламентноеЗадание**. При этом в свойстве **Наименование** необходимо указывать представление объекта, на основании которого создается регламентное задание. Например, есть рассылка отчетов (элемент справочника), расписание, которое было настроено в карточке рассылки и ее автор, тогда добавление на основании нее регламентного задания будет выглядеть так:

// Снимаем ограничение, что только администратор может создавать регламентные задания.  
УстановитьПривилегированныйРежим(Истина);  
Задание = РегламентныеЗадания.СоздатьРегламентноеЗадание(Метаданные.РегламентныеЗадания.РассылкаОтчетов);

ПараметрыЗадания = Новый Массив;  
ПараметрыЗадания.Добавить(РассылкаОтчетов);  
Задание.Параметры = ПараметрыЗадания;

Задание.ИмяПользователя = АвторРассылки;  
Задание.Использование = Истина;  
Задание.Наименование = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru = 'Рассылка отчетов: %1'"), СокрЛП(РассылкаОтчетов);  
Задание.Расписание = РасписаниеРассылки;   
Задание.Записать();

4.2. Если в конфигурации используется подсистема «Регламентные задания» БСП, то необходимо также запрещать интерактивное создание и запуск параметризованных регламентных заданий из формы **Регламентные и фоновые задания**. Для этого необходимо указать такое задание в процедуре **ПриОпределенииНастроекРегламентныхЗаданий** общего модуля **РегламентныеЗаданияПереопределяемый**. Например:

Настройка = Настройки.Добавить();  
Настройка.РегламентноеЗадание = Метаданные.РегламентныеЗадания.РассылкаОтчетов;  
Настройка.Параметризуется = Истина;

Также выполнить п.6.

5. Во избежание различных конфликтных ситуаций рекомендуется в копиях информационной базы автоматически блокировать все регламентные задания, обращающиеся к внешним ресурсам (рассылка почты, синхронизация данных с другими программами и т.п.). Например, если копия информационной базы была развернута для тестирования или передана в службу технической поддержки.

Если в конфигурации используется подсистема «Регламентные задания» БСП, то для этого необходимо перечислить такие задания в процедуре **ПриОпределенииНастроекРегламентныхЗаданий**общего модуля **РегламентныеЗаданияПереопределяемый**. Например:

Настройка = Настройки.Добавить();  
Настройка.РегламентноеЗадание = Метаданные.РегламентныеЗадания.РассылкаОтчетов;  
Настройка.РаботаетСВнешнимиРесурсами = Истина;

И выполнить п.6.

В этом случае при перемещении информационной базы администратору будет задан вопрос об отключении таких заданий.

6. Если регламентное задание попадает под требования, описанные в пунктах 3.1, 4.2, 5 и используется подсистема «Регламентные задания» БСП, то вначале процедур обработчиков таких заданий необходимо помещать вызов:

ОбщегоНазначения.ПриНачалеВыполненияРегламентногоЗадания(Метаданные.РегламентныеЗадания.<ИмяЗадания>);

Первый параметр при этом заполнять обязательно.

См. также

* [Настройка расписания регламентных заданий](https://its.1c.ua/db/content/v8std/src/100/500/i8100402.htm?_=1579516850)
* [Запуск регламентных заданий](https://its.1c.ua/db/content/v8std/src/100/500/i8100539.htm?_=1579516850)
* [Регламентные задания: требования по локализации](https://its.1c.ua/db/content/v8std/src/900/i8100767.htm?_=1579516850)
* [Ограничения на регламентные задания при работе в режиме сервиса](https://its.1c.ua/db/content/v8std/src/100/500/i8100760.htm?_=1579516850)

### Настройка расписания регламентных заданий

#std402

Область применения: управляемое приложение, обычное приложение.

При разработке регламентных заданий необходимо выбирать время и интервал запуска, исходя из прикладного назначения регламентных заданий, а также руководствуясь соображением, что частое выполнение регламентных заданий может негативно влиять на производительность сервера приложений **1С:Предприятие**:

* регламентное задание не должно выполняться чаще, чем это нужно с прикладной точки зрения;
* с точки зрения оптимальной загрузки сервера приложений для большинства регламентных заданий нормальным является интервал выполнения заданий в 1 день;
* исключения могут составлять случаи, когда критичным является частое выполнение заданий с прикладной точки зрения, например, для поддержания актуальности данных за короткий период;
* ни в каких случаях не следует задавать периодичность выполнения регламентных заданий меньше одной минуты;
* периодичность выполнения частых (с периодичностью менее одного дня) регламентных заданий должна быть сбалансирована со временем выполнения задания: например, если типичное время выполнения 20 секунд, то периодичность раз в минуту, скорее всего, избыточна;
* выполнение ресурсоемких регламентных операций необходимо по возможности переносить на время минимальной загрузки сервера приложений **1С:Предприятие**. Например, в нерабочее время или на выходные дни;
* несколько различных ресурсоемких регламентных заданий лучше "разносить" по времени, исходя из ожидаемого времени их выполнения.

См. также

* [Предопределенные регламентные задания](https://its.1c.ua/db/content/v8std/src/100/500/i8100540.htm?_=1579516850)
* [Запуск регламентных заданий](https://its.1c.ua/db/content/v8std/src/100/500/i8100539.htm?_=1579516850)

### Запуск регламентных заданий

#std539 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, обычное приложение.

1. Рекомендуется предоставлять пользователям альтернативную возможность по выполнению регламентных заданий вручную. Например, предлагать «по кнопке» выполнить обработку данных, обычно выполняемую регламентным заданием в фоне. Это вызвано тем соображением, что работа системы не должна зависеть от автоматического выполнения регламентных заданий. В частности:

* выполнение регламентных заданий может быть осознанно выключено на кластере серверов 1С:Предприятия;
* в отличие от клиент-серверного режима работы 1С:Предприятия версии 8.2 и ранее, в котором регламентные и фоновые задания выполняются на сервере, в файловом режиме отсутствовала возможность по их автоматическому выполнению.

В зависимости от специфики регламентных заданий, различается способ их запуска.

1.1. В случае если **регламентное задание изменяет в системе некоторые данные**, которые необходимы определенному бизнес-процессу или выводятся в конкретном «рабочем месте» (форме), то в таких «рабочих местах» дополнительно рекомендуется размещать команду для выполнения этого действия. Например:

* в форме для поиска в данных рекомендуется вывести дату актуальности индекса, если он не актуален, и команду «Обновить»;
* в списке входящих писем указано, когда они последний раз принимались, и имеется команда «Получить почту»;
* в рабочем месте ответственного за партионный учет указано, на какой момент времени проводилось последний раз распределение по партиям, и команда «Выполнить» для распределения по партиям.

Такие рабочие места должны информировать пользователя о дате актуальности представленных данных и команду для их обновления или обработки (которая выполняет то же действие, что и регламентное задание). Команда должна быть доступна только пользователям с необходимыми для ее выполнения правами.

Пример ручного запуска задания по очистке устаревших версий объектов:

&НаСервере  
Процедура ЗапуститьРегламентноеЗадание()  
   ИмяМетода = Метаданные.РегламентныеЗадания.ОчисткаУстаревшихВерсийОбъектов.ИмяМетода;

   // Проверка, выполняется ли фоновое задание по очистке устаревших версий.  
   Отбор = Новый Структура;  
   Отбор.Вставить("ИмяМетода", ИмяМетода);  
   Отбор.Вставить("Состояние", СостояниеФоновогоЗадания.Активно);  
   ФоновыеЗаданияОчистки = ФоновыеЗадания.ПолучитьФоновыеЗадания(Отбор);  
   Если ФоновыеЗаданияОчистки.Количество() = 0 Тогда  
      НаименованиеФоновогоЗадания = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(  
        НСтр("ru = 'Запуск вручную: %1'"), РегламентноеЗаданиеМетаданные.Синоним);  
      ФоновыеЗадания.Выполнить(ИмяМетода,,, НаименованиеФоновогоЗадания);  
   КонецЕсли;  
КонецПроцедуры

1.2. Если **работа регламентного задания оказывает влияние на данные, отображаемые в заранее неизвестном количестве рабочих мест, или влияет на информационную базу в целом**, то не представляется возможным выделить какое-то одно рабочее место для размещения команды запуска всех таких заданий. Примеры регламентных заданий, не «привязанных» к конкретным рабочим местам:

* обновление и перестроение агрегатов;
* установка периода рассчитанных итогов.

Результат работы таких регламентных заданий оказывает влияние сразу на множество внутренних и внешних отчетов системы, которые опираются на итоги и агрегаты.

В этом случае рекомендуется заводить отдельное рабочее место для выполнения таких регламентных заданий. При использовании в конфигурации Библиотеки стандартных подсистем такое рабочее место уже входит в состав подсистемы «Регламентные задания» (форма «Регламентные и фоновые задания»).

1.3. В тех же случаях когда **регламентное задание не изменяет данные в системе**, а формирует различные отчеты или рассылки из нее, также рекомендуется предусматривать отдельное рабочее место для выполнения таких регламентных заданий.

Примеры регламентных заданий, которые не меняют данные в базе:

* рассылка по почтовым адресатам информации об ошибках в журнале регистрации;
* рассылка информации о новых/просроченных задачах;
* периодический запуск внешних обработок для рассылки отчетов.

При использовании в конфигурации Библиотеки стандартных подсистем такое рабочее место уже входит в состав подсистемы «Регламентные задания» (форма «Регламентные и фоновые задания»).

2. Для администраторов информационных баз действует рекомендация: на период выполнения обновления ИБ блокировать работу регламентных заданий. Однако если обновление выполняет неподготовленный пользователь, в особенности, в файловом режиме работы, то рекомендуется дополнительно предусмотреть следующие меры:

* в файловом режиме работы, при неудачной попытке установки монопольного режима для обновления данных ИБ предлагать автоматически блокировать работу регламентных заданий (перезапуск программы с ключом командной строки **/AllowExecuteScheduledJobs -Off**);
* в начале кода обработчиков регламентных заданий проверять режим работы и прерывать работу регламентного задания с помощью вызова исключения, если обновление ИБ еще не завершено.

При использовании в конфигурации **Библиотеки стандартных подсистем** первая рекомендация реализована в подсистеме «Обновление версии ИБ», а для выполнения второй предусмотрена процедура **ПриНачалеВыполненияРегламентногоЗадания** общего модуля **ОбщегоНазначения**, вызов которой необходимо размещать в начале кода обработчиков регламентных заданий.

См. также

* [Настройка расписания регламентных заданий](https://its.1c.ua/db/content/v8std/src/100/500/i8100402.htm?_=1579516850)
* [Предопределенные регламентные задания](https://its.1c.ua/db/content/v8std/src/100/500/i8100540.htm?_=1579516850)
* [Ограничения на регламентные задания при работе в режиме сервиса](https://its.1c.ua/db/content/v8std/src/100/500/i8100760.htm?_=1579516850)

### Ограничения на регламентные задания при работе в режиме сервиса

#std760

Область применения: управляемое приложение.

1. В прикладных решениях, ориентированных на работу в режиме сервиса по Технологии 1cFresh, не должно быть регламентных заданий, которые включены в состав любого из разделителей. Это ограничение обусловлено тем, что при большом количестве областей данных в одной информационной базе разделенные регламентные задания могут вызвать перегрузку рабочих процессов, обслуживающих данную информационную базу, и серьезно затруднить работу пользователей сервиса.

2. Если требуется обеспечить регулярное выполнение определенного программного кода в каждой области данных разделенной информационной базы, необходимо использовать подсистему БСП «Очередь заданий», либо разработать аналогичный механизм очереди заданий самостоятельно.  
  
Например, требуется добавить в конфигурацию регламентное задание **ПроверкаЦен**, которое должно по расписанию выполнять в каждой области проверку прайс-листов, сопоставлять цены с динамикой валютных курсов, и при необходимости формировать некие сообщения для пользователей.  
  
Неправильно:  
  
Добавить в конфигурацию регламентное заданий **ПроверкаЦен** и включить его в состав общего реквизита **ОбластьДанныхОсновныеДанные**.  
  
Правильно:

* Реализовать прикладную функциональность проверки. Предположим, это будет процедура **ПроверитьЦены** модуля **УправлениеЦенами**.
* Добавить в конфигурацию предопределенное неразделенное регламентное задание **ПроверкаЦен**. Установить в качестве обработчика процедуру **УправлениеЦенами.ПроверитьЦены**.
* Добавить в общий модуль **ОчередьЗаданийПереопределяемый** следующий программный код:

Процедура ПриПолученииСпискаШаблонов(Шаблоны) Экспорт  
        Шаблоны.Добавить(Метаданные.РегламентныеЗадания.ПроверкаЦен.Имя);  
КонецПроцедуры  
  
Процедура ПриОпределенииПсевдонимовОбработчиков(СоответствиеИменПсевдонимам) Экспорт  
        СоответствиеИменПсевдонимам.Вставить(Метаданные.РегламентныеЗадания.ПроверкаЦен.ИмяМетода);  
КонецПроцедуры

3. Единственным исключением является ситуация, когда регламентное задание обязательно должно выполняться от имени определенного пользователя. Например, может потребоваться, чтобы при выполнении задания учитывались установленные для пользователя ограничения доступа к данным. В этом случае разделение регламентного задания допускается, но такое задание обязательно должно быть включено в состав всех разделителей, определенных в конфигурации.

4. В прикладных решениях, ориентированных на работу в режиме сервиса по Технологии 1cFresh, не должно быть участков, где из программного кода напрямую выполняется управление регламентными заданиями. Для управления регламентными заданиями необходимо использовать программный интерфейс БСП, реализованный в модуле **РегламентныеЗаданияСервер**.  
  
Неправильно:

// Ищем задание по наименованию.  
Отбор = Новый Структура();  
Отбор.Вставить(“Метаданные”, “ПроверкаЦен”);  
Задания = РегламентныеЗадания.ПолучитьРегламентныеЗадания(Отбор);

// Проверяем, что задание найдено.  
Если Задания.Количество() <> 1 Тогда  
 // Запись в журнал ошибки опущена.  
 Возврат;  
КонецЕсли;

// Включаем найденное задание.  
НашеЗадание = Задания[0];  
НашеЗадание.Использование = Истина;  
НашеЗадание.Записать();

Правильно:

// Ищем задание по наименованию.  
Отбор = Новый Структура();  
Отбор.Вставить(“Метаданные”, “ПроверкаЦен”);  
Задания = РегламентныеЗаданияСервер.НайтиЗадания(Отбор);

// Проверяем, что задание найдено.  
Если Задания.Количество() <> 1 Тогда  
 // Запись в журнал ошибки опущена.  
 Возврат;  
КонецЕсли;

// Включаем найденное задание.  
НашеЗадание = Задания[0];  
Параметры = Новый Структура();  
Параметры.Вставить(“Использование”, Истина);  
РегламентныеЗаданияСервер.ИзменитьЗадание(НашеЗадание.УникальныйИдентификатор, Параметры);

5. Следует учитывать, что подсистема «Очередь заданий» не гарантирует выполнение регламентного задания в точном соответствии с указанным расписанием. Точность соблюдения расписания зависит от общего количества запланированных заданий, длительности их выполнения и количества исполняющих потоков (регулируется константой «Максимальное количество исполняющихся фоновых заданий»).  
Рекомендуется в общем случае при работе в режиме сервиса не предоставлять пользователям возможность настройки расписания регламентных заданий.

# [Реализация обработки данных](https://its.1c.ua/db/v8std/browse/13/-1/26)

## Работа с запросами

### Оформление текстов запросов

#std437

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Все ключевые слова языка запросов пишутся заглавными буквами.

|  |
| --- |
| Методическая рекомендация (полезный совет)  2. Рекомендуется указывать и необязательные конструкции запроса, прежде всего - явно назначать псевдонимы полям, в целях повышения наглядности текста запроса и "устойчивости" использующего его кода. Например, если в алгоритме используется запрос с полем, объявленным как  Касса.Валюта  при изменении имени реквизита нужно будет также изменить и код, осуществляющий обращение по имени свойства **Валюта** к выборке из результата запроса. Если же поле будет объявлено как  Касса.Валюта КАК Валюта  то изменение имени реквизита приведет только к изменению текста запроса.  2а. Особенно внимательно следует относиться к автоматически присваиваемым псевдонимам для полей – реквизитов других полей, типа "... Касса.Валюта.Наименование...". В приведенном выше примере поле получит автоматический псевдоним **ВалютаНаименование**, а не **Наименование**.  2б. Следует обязательно указывать ключевое слово **КАК** перед псевдонимом поля источника. |

3. Текст запроса должен быть структурирован, не следует писать запрос в одну строку, даже короткий. Текст запроса должен быть нагляден, поскольку это существенно улучшает его понимание другими разработчиками.  
  
4. В запросы, сложные для понимания, в которых используются вложенные запросы, объединения или соединения рекомендуется вставлять комментарии. Комментарии, например, могут объяснять для получения каких данных используется та или иная таблица в соединении или объединении.

При этом необходимо иметь в виду, что при использовании конструктора запросов, все комментарии в запросе удаляются автоматически без предупреждения.  
  
5. При создании объекта **Запрос** рекомендуется указывать комментарии, для получения какой информации или каких иных целей будет использован данный запрос.  
  
6.1 При программной "сборке" текста запроса рекомендуется комментировать все этапы его сборки.

6.2. Нужно стараться, чтобы каждая часть формируемого запроса могла быть открыта с помощью конструктора запросов

* это позволяет осуществить экспресс-проверку корректности синтаксиса запроса
* это упрощает разработку и сопровождение кода конфигурации, в том числе сторонними разработчиками

Типичные случаи программной модификации текста запроса

#### Изменение имени поля выборки или таблицы

Неправильно

ТекстЗапроса =

"ВЫБРАТЬ  
| Номенклатура.Наименование  КАК Наименование ,  
| Номенклатура. " + ИмяПоляКод + " КАК КодАртикул  
|ИЗ  
| Справочник.Номенклатура КАК Номенклатура";

Правильно

ТекстЗапроса =  
"ВЫБРАТЬ  
| Номенклатура.Наименование  КАК Наименование,  
| &ИмяПоляКод  КАК КодАртикул  
|ИЗ  
| Справочник.Номенклатура КАК Номенклатура ;

ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "&ИмяПоляКод ", "Номенклатура." + ИмяПоляКод);

или аналогично для имени таблицы

ТекстЗапроса =

"ВЫБРАТЬ  
| ТаблицаСправочника.Наименование  КАК Наименование,  
| ТаблицаСправочника.Код  КАК Код  
|ИЗ  
| &ТаблицаСправочника КАК ТаблицаСправочника";  
ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "&ТаблицаСправочника", "Справочник." + ИмяСправочника);

или еще один вариант для имени таблицы

ТекстЗапроса =

"ВЫБРАТЬ  
| Номенклатура.Наименование  КАК НаименованиеТовара ,  
| ЕСТЬNULL(ТаблицаОстатков.ВНаличииОстаток,0) КАК ОстатокТовара  
|ИЗ  
| Справочник.Номенклатура КАК Номенклатура  
| ЛЕВОЕ СОЕДИНЕНИЕ #ТаблицаОстатков КАК ТаблицаОстатков  
| ПО Номенклатура.Ссылка= ТаблицаОстатков.Номенклатура";  
  
Если ИспользуетсяАдресноеХранение Тогда  
 ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "#ТаблицаОстатков", "РегистрНакопления.ТоварыВЯчейках.Остатки");  
Иначе  
 ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "#ТаблицаОстатков", "РегистрНакопления.ТоварыНаСкладах.Остатки");  
КонецЕсли;

#### Конкатенация нескольких текстов запросов в пакет

Неправильно

ТекстЗапроса = " ";

Если ИспользоватьУпаковки Тогда

ТекстЗапроса =

"ВЫБРАТЬ  
| Упаковки.Ссылка КАК Ссылка  
|ИЗ  
| Справочник.Упаковки КАК Упаковки;  
|/////////////////////////////////////////////////////////////  
|";

КонецЕсли;

ТекстЗапроса = ТекстЗапроса +  
"ВЫБРАТЬ  
| Номенклатура.Ссылка КАК Ссылка  
|ИЗ  
| Справочник. Номенклатура КАК Номенклатура";

Правильно

ТекстЗапроса = " ";

Если ИспользоватьУпаковки Тогда

ТекстЗапроса =

"ВЫБРАТЬ  
| Упаковки.Ссылка КАК Ссылка  
|ИЗ  
| Справочник.Упаковки КАК Упаковки";

ТекстЗапроса = ТекстЗапроса +  
"  
|;  
|/////////////////////////////////////////////////////////////  
|";

КонецЕсли;

ТекстЗапроса = ТекстЗапроса +  
"ВЫБРАТЬ  
| Номенклатура.Ссылка КАК Ссылка  
|ИЗ  
| Справочник.Номенклатура КАК Номенклатура";

Или

Разделитель =  
"  
|;  
|/////////////////////////////////////////////////////////////  
|";

ТекстыЗапросовПакета = Новый Массив;

ТекстЗапроса =  
"ВЫБРАТЬ  
| Упаковки.Ссылка КАК Ссылка  
|ИЗ  
| Справочник.Упаковки КАК Упаковки";

ТекстыЗапросовПакета.Добавить(ТекстЗапроса);

ТекстЗапроса =  
"ВЫБРАТЬ  
| Номенклатура.Ссылка КАК Ссылка  
|ИЗ  
| Справочник.Номенклатура КАК Номенклатура ";

ТекстыЗапросовПакета.Добавить(ТекстЗапроса);  
ТекстЗапроса = СтрСоединить(ТекстыЗапросовПакета, Разделитель);

См. также

* [Запросы, динамические списки и отчеты на СКД: требования по локализации](https://its.1c.ua/db/content/v8std/src/900/i8100762.htm?_=1579516850)

### Многократное выполнение однотипных запросов

#std436

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Рекомендуется получать все необходимые однотипные данные одним запросом, вместо выполнения серии запросов.  
  
Правильно:

// БанкиДляОбработки - содержит массив банков, счета в которых необходимо обработать  
  
ОбщийЗапрос = Новый Запрос("  
  |ВЫБРАТЬ  
  | БанковскиеСчета.Ссылка КАК Счет  
  |ИЗ  
  | Справочник.БанковскиеСчета КАК БанковскиеСчета  
  |ГДЕ  
  | БанковскиеСчета.Банк В(&БанкиДляОбработки)");  
  
ОбщийЗапрос.УстановитьПараметр("БанкиДляОбработки", БанкиДляОбработки);  
ВыборкаСчетов = ОбщийЗапрос.Выполнить().Выбрать();  
Пока ВыборкаСчетов.Следующий() Цикл  
  ОбработатьСчетаВБанке(ВыборкаСчетов.Счет);  
КонецЦикла;

Неправильно:

// БанкиДляОбработки - содержит массив банков, счета в которых необходимо обработать  
  
ЧастныйЗапрос = Новый Запрос("  
  |ВЫБРАТЬ  
  | БанковскиеСчета.Ссылка КАК Счет  
  |ИЗ  
  | Справочник.БанковскиеСчета КАК БанковскиеСчета  
  |ГДЕ  
  | БанковскиеСчета.Банк = &Банк");  
  
Для каждого Банк Из БанкиДляОбработки Цикл  
  ЧастныйЗапрос.УстановитьПараметр("Банк", Банк);  
  ВыборкаСчетов = ЧастныйЗапрос.Выполнить().Выбрать();  
  Пока ВыборкаСчетов.Следующий() Цикл  
    ОбработатьСчетаВБанке(ВыборкаСчетов.Счет);  
  КонецЦикла;  
КонецЦикла;

### Проверка на пустой результат выполнения запроса

#std438

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Проверку того, что результат выполнения запроса не содержит строк следует выполнять с помощью метода **Пустой**. Поскольку на получение выборки из результата запроса (выгрузка его в таблицу значений) будет затрачиваться дополнительное время.  
  
Неправильно:

Выборка = Запрос.Выполнить().Выбрать();  
Если Выборка.Следующий() Тогда  
  Возврат Истина;  
Иначе  
  Возврат Ложь;  
КонецЕсли;

Правильно:

Возврат НЕ Запрос.Выполнить().Пустой()

|  |
| --- |
| Методическая рекомендация (полезный совет)  2. В то же время если требуется выбрать (или выгрузить) результат запроса, то предварительный вызов метода **Пустой** не требуется. Например, вместо:  РезультатЗапроса = Запрос.Выполнить(); Если НЕ РезультатЗапроса.Пустой() Тогда // избыточный вызов   Выборка = РезультатЗапроса.Выбрать();    Пока Выборка.Следующий() Цикл   ...  правильно:  Выборка = Запрос.Выполнить().Выбрать(); Пока Выборка.Следующий() Цикл ... |

### Ограничение на использование конструкции "ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ" в запросах

#std435

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1.1. При разработке текстов запросов следует иметь в виду, что при работе в клиент-серверном варианте, когда в качестве СУБД используется **PostgreSQL**, производительность выполнения запросов с конструкцией**ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** значительно снижается. В особенности это касается случаев, когда в запросе встречаются две и более таких конструкций.  Поэтому в общем случае не рекомендуется использовать конструкцию **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** в запросах. И в тех случаях, где это возможно, рекомендуется переписать текст исходного запроса без использования этой конструкции.  Например, следующий запрос:  ВЫБРАТЬ  ЕСТЬNULL(ПланПродаж.Номенклатура, ФактическиеПродажи.Номенклатура) КАК Номенклатура,  ЕСТЬNULL(ПланПродаж.Сумма, 0) КАК СуммаПлан,  ЕСТЬNULL(ФактическиеПродажи.Сумма, 0) КАК СуммаФакт ИЗ  ПланПродаж КАК ПланПродаж   ПОЛНОЕ СОЕДИНЕНИЕ ФактическиеПродажи КАК ФактическиеПродажи   ПО ПланПродаж.Номенклатура = ФактическиеПродажи.Номенклатура  может быть реализован без конструкции  **ПОЛНОЕ [ВНЕШНЕЕ] СОЕДИНЕНИЕ** следующим образом:  ВЫБРАТЬ  ПланФактПродаж.Номенклатура КАК Номенклатура,  СУММА(ПланФактПродаж.СуммаПлан) КАК СуммаПлан,  СУММА(ПланФактПродаж.СуммаФакт) КАК СуммаФакт ИЗ  (ВЫБРАТЬ   ПланПродаж.Номенклатура КАК Номенклатура,   ПланПродаж.Сумма КАК СуммаПлан,   0 КАК СуммаФакт  ИЗ   ПланПродаж КАК ПланПродаж    ОБЪЕДИНИТЬ ВСЕ    ВЫБРАТЬ   ФактическиеПродажи.Номенклатура,   0,   ФактическиеПродажи.Сумма  ИЗ   ФактическиеПродажи КАК ФактическиеПродажи) КАК ПланФактПродаж  СГРУППИРОВАТЬ ПО  ПланФактПродаж.Номенклатура  1.2. Исключение составляют случаи, когда текст исходного запроса не может быть переписан без использования конструкции **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** по объективным причинам. Следует иметь в виду, что при выполнении данной конструкции на СУБД **PostgreSQL** она автоматически заменяется платформой **1С:Предприятие** на эквивалентную, которая может быть исполнена в СУБД **PostgreSQL**. При этом сохраняются все атрибуты запроса, такие как модификаторы **ПЕРВЫЕ**, **РАЗЛИЧНЫЕ**, а также **УПОРЯДОЧИТЬ ПО**. В таких случаях не следует "механически" заменять конструкцию **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** только с той целью, чтобы от нее избавиться в тексте запроса. |

2. Не допускается одновременно использовать конструкцию **ПОЛНОЕ СОЕДИНЕНИЕ** и обращение к табличным частям из раздела **ВЫБРАТЬ**.

Данное требование продиктовано особенностью выполнения подобных запросов на СУБД **PostgreSQL** и необходимостью переносимости прикладных решений на эту СУБД.

См. также

* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1579516850)

### Использование ключевых слов "ОБЪЕДИНИТЬ" и "ОБЪЕДИНИТЬ ВСЕ" в запросах

#std434

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

В общем случае, при объединении в запросе результатов нескольких запросов следует использовать конструкцию **ОБЪЕДИНИТЬ ВСЕ**, а не **ОБЪЕДИНИТЬ**. Поскольку во втором варианте, при объединении запросов полностью одинаковые строки заменяются одной, на что затрачивается дополнительное время, даже в случаях, когда одинаковых строк в запросах заведомо быть не может.

Исключением являются ситуации, когда выполнение замены нескольких одинаковых строк одной является необходимым условием выполнения запроса.  
  
Правильно:

ВЫБРАТЬ  
ПоступлениеТоваровУслуг.Ссылка  
ИЗ  
Документ.ПоступлениеТоваровУслуг КАК ПоступлениеТоваровУслуг  
  
ОБЪЕДИНИТЬ ВСЕ  
  
ВЫБРАТЬ  
РеализацияТоваровУслуг.Ссылка  
ИЗ  
Документ.РеализацияТоваровУслуг КАК РеализацияТоваровУслуг

Неправильно:

ВЫБРАТЬ  
ПоступлениеТоваровУслуг.Ссылка  
ИЗ  
Документ.ПоступлениеТоваровУслуг КАК ПоступлениеТоваровУслуг  
  
ОБЪЕДИНИТЬ  
  
ВЫБРАТЬ  
РеализацияТоваровУслуг.Ссылка  
ИЗ  
Документ.РеализацияТоваровУслуг КАК РеализацияТоваровУслуг

### Упорядочивание результатов запроса

#std412

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Если алгоритм обработки результатов запроса зависит от порядка записей в запросе или если результат обработки запроса в той или иной форме представляется пользователю, то в тексте запроса следует использовать предложение **УПОРЯДОЧИТЬ ПО**. В отсутствие выражения **УПОРЯДОЧИТЬ ПО** невозможно сделать никаких предположений о том, в каком порядке будут представлены записи в результатах запроса.

Типичные примеры проблем, которые могут возникать (даже при работе на одной и той же СУБД в непредсказуемые моменты времени):

* разная последовательность строк табличной части при заполнении по результатам запроса;
* разный порядок вывода данных (строк, колонок) в отчетах;
* разное заполнение движений документа по результатам запроса (\*).

Вероятность возникновения разных результатов при выполнении одинаковых действий повышается

* при переносе информационной базы на другую СУБД
* при смене версии СУБД
* при изменении параметров СУБД

\* Примечание: упорядочивание результатов запросов, по которым формируются движения, оправдано только в том случае, если упорядочивание является частью алгоритма формирования движений (например, списание остатков партий товаров по FIFO). В остальных случаях упорядочивать записи не следует, так как дополнительное упорядочивание будет создавать избыточную нагрузку на СУБД.

1.2. При сортировке по полю запроса, которое может потенциально содержать NULL, следует учитывать, что в разных СУБД порядок сортировки по этому полю может отличаться.

**Неправильно:**

ВЫБРАТЬ  
  СправочникНоменклатура.Ссылка КАК НоменклатураСсылка,  
  ЗапасыОстатки.КоличествоОстаток КАК КоличествоОстаток  
ИЗ  
  Справочник.Номенклатура КАК СправочникНоменклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Запасы.Остатки КАК ЗапасыОстатки  
    ПО (ЗапасыОстатки.Номенклатура = СправочникНоменклатура.Ссылка)

УПОРЯДОЧИТЬ ПО  
  КоличествоОстаток

**Правильно:**

ВЫБРАТЬ  
  СправочникНоменклатура.Ссылка КАК НоменклатураСсылка,  
  ЕСТЬNULL(ЗапасыОстатки.КоличествоОстаток, 0) КАК КоличествоОстаток  
ИЗ  
  Справочник.Номенклатура КАК СправочникНоменклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.Запасы.Остатки КАК ЗапасыОстатки  
    ПО (ЗапасыОстатки.Номенклатура = СправочникНоменклатура.Ссылка)

УПОРЯДОЧИТЬ ПО  
  КоличествоОстаток

См. также: [*Особенности работы с различными СУБД*](https://its.1c.ua/db/v8310doc/bookmark/dev/TI000001285)

1.3. Если результаты запроса должны тем или иным образом отображаться пользователю, то

* упорядочивать результаты таких запросов необходимо по полям примитивных типов;
* упорядочивание по полям ссылочных типов нужно заменять на упорядочивание по строковым представлениям этих полей.

В противном случае порядок следования строк будет выглядеть для пользователя случайным (необъяснимым).

См. также: [*Сортировка строк таблиц значений*](https://its.1c.ua/db/content/v8std/src/300/300/i8100452.htm?_=1579516850)

1.4. Отсутствие предложения **УПОРЯДОЧИТЬ ПО** оправдано только в тех случаях, когда

* алгоритм обработки результатов запроса не рассчитывает на определенный порядок записей
* результат обработки выполненного запроса не показывается пользователю
* результат запроса - заведомо одна запись

В таких случаях рекомендуется не добавлять предложение **УПОРЯДОЧИТЬ ПО** в текст запроса, так как это приводит к дополнительным затратам времени при выполнении запроса.

#### Совместное использование с конструкцией РАЗЛИЧНЫЕ

2. Если в запросе используется конструкция **РАЗЛИЧНЫЕ**, упорядочивание следует выполнять только по полям, включенным в выборку (в секции **ВЫБРАТЬ**).   
  
Данное требование связано со следующей особенностью выполнения запросов: в поля выборки неявно включаются поля упорядочивания, что в свою очередь может привести к появлению в результате запроса нескольких строк с одинаковыми значениями полей выборки.

#### Ограничения на использование конструкции АВТОУПОРЯДОЧИВАНИЕ

3. Использование конструкции **ПЕРВЫЕ** совместно с конструкцией **АВТОУПОРЯДОЧИВАНИЕ** запрещено.

В остальных случаях конструкцию **АВТОУПОРЯДОЧИВАНИЕ** также не рекомендуется использовать, так как разработчик не контролирует, какие именно поля будут использованы для упорядочивания. Применение такой конструкции оправдано только в тех случаях, когда получаемый порядок записей не важен, но при этом он должен быть одинаковым в не зависимости от применяемой СУБД.

Причины использования конструкции **АВТОУПОРЯДОЧИВАНИЕ** следует указывать в комментарии, размещенном непосредственно перед текстом запроса.

### Округление результатов арифметических операций в запросах

#std535

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1. Если в операции деления заранее известны порядки числителя и знаменателя, то следует по возможности избегать выполнения деления числа заведомого маленького порядка на число заведомо большого порядка. Например, вместо:  0.02 / 28346 \* 9287492  правильно:  0.02 \* 9287492 / 28346  2. При выполнении арифметических операций в запросах к базе данных платформа **1С:Предприятия** поддерживает точность вычислений до 8 разрядов дробной части. Однако, из-за особенностей работы различных СУБД в некоторых ситуациях точность результатов может отличаться от 8. Более подробно о вычислении разрядности результатов можно почитать в статье ИТС [Разрядность результатов выражений и агрегатных функций в языке запросов](http://its.1c.ru/db/metod81#content:2665:1).  Если точность результата выполнения запроса к базе данных, содержащего   * арифметические операции деления, * агрегатные функции **СРЕДНЕЕ**, * арифметические операции умножения, если каждый из множителей может иметь дробную часть,   различается на различных СУБД, то рекомендуется к операндам и/ или результатам этих операций применять оператор явного приведения разрядности и точности числовых данных:  ВЫРАЗИТЬ(... КАК Число(m, n))  Оператор **ВЫРАЗИТЬ** следует применять к операндам, если на какой-нибудь СУБД точность получаемого результата недостаточна. Например, требуется 10 разрядов после запятой, а получается 6.  При этом указанная общая разрядность операндов должна быть минимальной, но не меньше той, которая достаточна для представления значений каждого из операндов. Неоправданное завышение разрядности может привести к потере точности последующих вычислений и несколько снизить скорость выполнения запроса.  Важно иметь в виду, что на разных СУБД имеются различные ограничения на максимальную разрядность десятичных чисел. Самое жесткое ограничение - это 31 разряд в целой и дробной частях. Чем меньшее значение разрядности будет указано для операндов, тем выше сможет быть точность результата. Например, если в результате требуется не менее 10 разрядов дробной части, первый операнд заведомо помещается в 15 разрядов целой части, а второй операнд заведомо помещается в 5 знаков целой части, то выражение может быть записано так:  ВЫБРАТЬ ВЫРАЗИТЬ(Таблица.Множитель \* Таблица.Числитель КАК Число(25,10)) / ВЫРАЗИТЬ(Таблица.Знаменатель КАК Число(15,10)) КАК Результат ИЗ Таблица КАК Таблица  Оператор **ВЫРАЗИТЬ** следует применять к результату, если точность вычислений на всех СУБД достаточна, но на некоторых она больше, а на других меньше. При этом указанная общая разрядность результата должна быть минимальной, но не меньше той, которая достаточна для представления значений результата. Если в приведенном примере известно, что **Знаменатель** не может быть меньше 0.00001, то для представления результата достаточно 20 разрядов целой части. В этом случае выражение может быть записано так:  ВЫБРАТЬ ВЫРАЗИТЬ(Таблица.Множитель \* Таблица.Числитель / Таблица.Знаменатель КАК Число(30,10)) КАК Результат ИЗ Таблица КАК Таблица  Иногда может быть целесообразно выполнить приведения к требуемой точности как операндов, так и результата. Например:  ВЫБРАТЬ ВЫРАЗИТЬ(ВЫРАЗИТЬ(Таблица.Множитель \* Таблица.Числитель КАК Число(25,10)) / ВЫРАЗИТЬ(Таблица.Знаменатель КАК Число(15,10)) КАК Число(30,10)) КАК Результат ИЗ Таблица КАК Таблица |

### Особенности использования в запросах оператора ПОДОБНО

#std726

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При использовании в тексте запроса оператора ПОДОБНО допустимо использовать только константные строковые литералы или параметры запроса. Запрещается формировать строку шаблона при помощи вычислений, использовать конкатенацию строк средствами языка запросов. Например:

Допустимо:

Реквизит ПОДОБНО "123%"

Недопустимо:

Реквизит ПОДОБНО "123" + "%"  
Реквизит ПОДОБНО Таблица.Шаблон

2. Запросы, в которых управляющие символы шаблона оператора ПОДОБНО находятся в полях запроса или в вычисляемых выражениях, по-разному интерпретируются на различных СУБД. Запрос, успешно выполняющийся, например, при работе с файловой базой, может возвращать неверные результаты при работе в режиме клиент-сервера. Подобные выражения необходимо переформулировать.

Например, вместо:

Запрос = Новый Запрос("

|ВЫБРАТЬ

| Товары.Ссылка

|ИЗ

| Справочник.Товары КАК Товары

|ГДЕ

| Товары.СтранаПроисхождения.Наименование ПОДОБНО &ШаблонНазванияСтраны + "\_"

|");

Запрос.УстановитьПараметр("ШаблонНазванияСтраны", "ЧА");

Необходимо использовать:

Запрос = Новый Запрос("

|ВЫБРАТЬ

| Товары.Ссылка

|ИЗ

| Справочник.Товары КАК Товары

|ГДЕ

| Товары.СтранаПроисхождения.Наименование ПОДОБНО &ШаблонНазванияСтраны

|");

Запрос.УстановитьПараметр("ШаблонНазванияСтраны", "ЧА\_");

Данное требование продиктовано необходимостью переносимости прикладных решений на различные СУБД.

См. также

* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1579516850)
* [Оформление текстов запросов](https://its.1c.ua/db/content/v8std/src/200/100/i8100437.htm?_=1579516850)

### Псевдонимы источников данных в запросах

#std758

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Псевдоним источника данных должен быть осмысленным, чтобы было понятным его назначение в данном контексте. Требования к псевдонимам источников схожи с требованиями к [именам переменных в коде.](https://its.1c.ua/db/content/v8std/src/300/100/i8100454.htm?_=1579516850)

* псевдонимы следует образовывать от терминов предметной области таким образом, чтобы было понятно, как источник данных будет использоваться в запросе;
* псевдонимы следует образовывать путем удаления пробелов между словами. При этом каждое слово в имени пишется с прописной буквы (например, **ТоварыНаСкладах**). Предлоги и местоимения из одной буквы также пишутся прописными буквами;
* псевдонимы запрещается начинать с подчеркивания;
* псевдонимы не должны состоять из одного символа.

**Неправильно:**

ВЫБРАТЬ  
  Таблица1.Ссылка КАК Товар,  
  ЕстьNULL(Таблица2.КоличествоОстаток, 0) КАК Остаток  
ИЗ  
  Справочник.Номенклатура КАК Таблица1  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки КАК Таблица2  
    ПО Таблица1.Ссылка = Таблица2.Номенклатура

**Правильно:**

ВЫБРАТЬ  
  ВсяНоменклатура.Ссылка КАК Товар,  
  ЕстьNULL(ОстаткиНаСкладах.КоличествоОстаток, 0) КАК Остаток  
ИЗ  
  Справочник.Номенклатура КАК ВсяНоменклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки КАК ОстаткиНаСкладах  
    ПО ВсяНоменклатура.Ссылка = ОстаткиНаСкладах.Номенклатура

В частности не рекомендуется использовать имена классов объектов метаданных ("Справочник", "Документ" и т.д.), т.к. обычно такой псевдоним не будет описывать назначение источника в конкретном запросе.

2. В ряде случаев при разработке универсальных механизмов, рассчитанных на работу с произвольными таблицами данных, или при написании универсальных запросов, когда вместо источника данных при исполнении кода подставляется имя конкретной таблицы, допустимо использование универсальных псевдонимов.

**Пример:**

"ВЫБРАТЬ  
  Таблица.Наименование КАК Наименование  
  Таблица.Код КАК Код  
ИЗ  
  &Таблица КАК Таблица";  
  
ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "&Таблица", "Справочник." + ИмяСправочника);

## Оптимизация запросов

### Общие требования по разработке оптимальных запросов

#std729 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Прежде чем перейти к более продвинутым методам оптимизации запросов, необходимо убедиться, что сам запрос – адекватен решаемой задаче.

1.1. Следует минимизировать объем выборки таким образом, чтобы выбирать ровно те данные, которые требуются для решения задачи.  
Например, если нужно получить значения конкретных полей, не следует выбирать все поля «на всякий случай» с помощью конструкции ВЫБРАТЬ \* ИЗ …  
Вместо выборки большого объема данных для их последующей обработки (свертка, сортировка, проведение вычислений и пр.) на сервере 1С:Предприятие, следует, в первую очередь, ответить на вопрос: «А есть ли возможность переложить эту работу на базу данных, чтобы получить уже готовый результат?»

1.2. Также в большинстве случаев, следует минимизировать и общее количество запросов к СУБД.

См. также: [*Многократное выполнение однотипных запросов*](https://its.1c.ua/db/content/v8std/src/200/100/i8100436.htm?_=1579516850).

2. С другой стороны, не следует пытаться любой ценой перенести выполнение задачи в СУБД. СУБД обычно оптимизирует и выполняет простые запросы более эффективно, чем сложные.

2.1. Следует рассмотреть альтернативные меры:

* по подготовке различных (более простых, частных) текстов запроса в зависимости от предусловий и значений параметров запроса – вместо отправки в СУБД одного большого универсального запроса;
* по более эффективной постобработке данных, выбранных запросом из СУБД, на стороне сервера 1С:Предприятия средствами встроенного языка.

2.2. При разработке запросов нужно быть уверенным, что они использует эффективные планы выполнения запросов. Для сложных запросов СУБД с высокой вероятностью выберет неправильный план выполнения запроса, что особенно актуально для СУБД DB2, PostgreSQL и Oracle.  
Поэтому не следует неоправданно усложнять запрос, в первую очередь:

* Не следует добавлять [вложенные запросы](https://its.1c.ua/db/content/v8std/src/200/200/i8100655.htm?_=1579516850) только для повышения читаемости.
* Избегать сложных условий соединения и в предложении ГДЕ, в особенности [содержащие подзапросы](https://its.1c.ua/db/content/v8std/src/200/200/i8100656.htm?_=1579516850) и конструкции ВЫБОР.
* Использовать в запросе минимально необходимое число таблиц. В зависимости от структуры таблиц, много может быть уже и 5-7 таблиц в одном запросе (время, затрачиваемое оптимизатором СУБД на анализ запроса, растет нелинейно, в итоге получается плохой план выполнения).

Для того чтобы узнать, какой план выполнения запроса выбран оптимизатором СУБД, можно воспользоваться консолью запросов, технологическим журналом или средствами СУБД. Как правило, запрос – сложный и будет плохо выполняться, если в скомпилированном плане выполнения запроса есть timeout warning, который означает, что оптимизатору СУБД не хватило времени на поиск наилучшего плана запроса.

См. также: [Запросы в динамических списках](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/800/i8100732.htm?_=1579516850)

### Несоответствие индексов и условий запроса

#std652 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Необходимо убедиться в том, что для всех условий, использованных в запросе, имеются подходящие индексы.

Условия используются в следующих секциях запроса:

* ВЫБРАТЬ … ИЗ … ГДЕ <условие>
* СОЕДИНЕНИЕ … ПО <условие>
* ВЫБРАТЬ … ИЗ <ВиртуальнаяТаблица>(, <условие>)
* ИМЕЮЩИЕ <условие>

Для каждого условия должен существовать подходящий индекс. Подходящим является индекс, удовлетворяющий следующим требованиям:

1. Индекс содержит все поля перечисленные в условии;
2. Эти поля находятся в самом начале индекса;
3. Эти поля идут подряд, то есть между ними не «вклиниваются» поля, не участвующие в условии запроса

1.2. Если в структуре базы данных отсутствует индекс, удовлетворяющий всем перечисленным условиям, то для получения результата СУБД будет вынуждена сканировать таблицу или один из ее индексов. Это приведет к увеличению времени выполнения запроса, а также к возможному снижению параллельности системы, поскольку возрастет количество установленных блокировок.

Требования к индексу связаны с физической структурой индекса в СУБД. Эта структура представляет собой дерево значений проиндексированных полей. На первом уровне дерева находятся значения первого поля индекса, на втором - второго и так далее. Такая структура позволяет достичь высокой эффективности при поиске по индексу. Кроме того, она гарантирует отсутствие деградации производительности индекса с ростом количества данных.

Однако, индекс такой структуры, очевидно, может быть использован только строго определенным образом. Сначала необходимо провести поиск по значению первого поля индекса, затем - второго и так далее. Если, например, условие по первому полю индекса не указано, то индекс уже не сможет обеспечить быстрый поиск. Если указано условие по нескольким первым полям индекса, а затем одно или несколько полей индекса не задано, то индекс может быть использован только частично.

2. При создании объекта метаданных **1С:Предприятие** автоматически создает индексы, которые должны подходить для работы большинства запросов.

Основные индексы, создаваемые **1С:Предприятием**:

* индекс по уникальному идентификатору (ссылке) для всех объектных сущностей (справочники, документы и т.д.);
* индекс по регистратору (ссылке на документ) для таблиц движений регистров, подчиненных регистратору;
* индекс по периоду и значениям всех измерений для итоговых таблиц регистров накопления;
* индекс по периоду, счету и значениям всех измерений для итоговых таблиц регистров бухгалтерии.

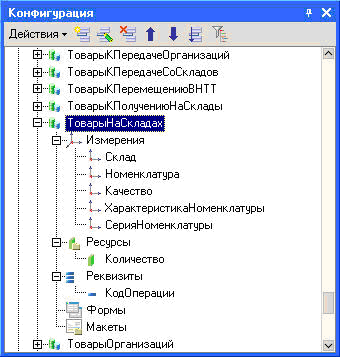
3. В тех случаях, когда автоматически созданных индексов недостаточно, можно дополнительно проиндексировать реквизиты объекта метаданных.

При этом реквизиты справочников и документов рекомендуется индексировать с дополнительным упорядочиванием. Такой индекс будет учитывать упорядочивание по основному представлению объекта, тем самым он будет эффективно использоваться, например, когда в списке установлен отбор по данному реквизиту, а сам список упорядочен по полям основного представления объекта.

Следует иметь в виду, что создание индекса ускоряет процесс поиска информации, но может несколько замедлить процесс ее изменения (добавления, редактирования и удаления). Поэтому индексы следует создавать осознанно и только в том случае, если точно известен запрос, для которого такой индекс необходим. Не следует создавать индексы "на всякий случай" или заведомо избыточные индексы. В частности:

* не следует дополнительно индексировать первое измерение регистра, поскольку для поиска по значению первого измерения подходит основной индекс таблицы итогов, который автоматически создаст платформа;
* не следует создавать индексы по низкоселектевным полям. Например, индексировать реквизит типа **Булево** имеет смысл, только если незначительная часть записей всегда будет иметь одно значение, и в запросах всегда выбираются записи по этому значению.

#### Примеры

В конфигурации описан регистр накопления ТоварыНаСкладах:

Платформа 1С:Предприятие автоматически создаст для таблицы остатков данного регистра индекс по периоду и всем измерениям в том порядке, в котором они перечислены в конфигураторе.

Рассмотрим несколько примеров запросов и проанализируем, смогут ли они оптимально выполняться при такой структуре данных.

#### Запрос 1

Запрос.Текст = "ВЫБРАТЬ  
| ТоварыНаСкладахОстатки.Склад,  
| ТоварыНаСкладахОстатки.Номенклатура,  
| ТоварыНаСкладахОстатки.Качество  
|ИЗ  
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Номенклатура = &Номенклатура) КАК ТоварыНаСкладахОстатки";

В данном случае нарушено требование 2 раздела 1.1. В условии отсутствует отбор по первому полю индекса (**Склад**). Такой запрос не сможет выполниться оптимально. Для его выполнения серверу СУБД придется перебирать все записи таблицы. Время выполнения этой операции напрямую зависит от количества записей в таблице остатков регистра и может быть очень большим (и будет увеличиваться с ростом количества данных).

Варианты оптимизации:

* Проиндексировать измерение **Номенклатура**
* Поставить измерение **Номенклатура** первым в списке измерений. Следует осторожно использовать этот метод. В конфигурации могут присутствовать другие запросы, которые могут замедлиться в результате этой перестановки.

#### Запрос 2

Запрос.Текст = "ВЫБРАТЬ  
| ТоварыНаСкладахОстатки.Склад,  
| ТоварыНаСкладахОстатки.Номенклатура,  
| ТоварыНаСкладахОстатки.Качество  
|ИЗ  
| РегистрНакопления.ТоварыНаСкладах.Остатки(  
| ,  
| Качество = &Качество  
| И Склад = &Склад) КАК ТоварыНаСкладахОстатки";

В данном случае нарушено требование 3 раздела 1.1. Между измерениями **Склад** и **Качество** в структуре регистра находится измерение **Номенклатура**, которое не задано в условии запроса. Этот запрос так же не сможет выполняться оптимально. При его выполнении СУБД выполнит поиск по первому полю индекса, но затем вынужденно просканирует некоторую его часть. Сканирование может привести к существенному увеличению времени выполнения запроса.

Варианты оптимизации:

* Добавить в запрос условие по измерению **Номенклатура**
* Убрать из запроса условие по измерению **Качество**
* Поменять местами измерения **Номенклатура** и **Качество**. Данную рекомендацию следует использовать, если временя выполнения запроса критическим образом сказывается на работы системы (например, запрос выполняется достаточно часто, или он выполняется не часто, но время его выполнения неприемлимо большое), т.к. это может замедлить выполнение других запросов, присутствующих в конфигурации.

#### Запрос 3

Запрос.Текст = "ВЫБРАТЬ  
| ТоварыНаСкладахОстатки.Склад,  
| ТоварыНаСкладахОстатки.Номенклатура,  
| ТоварыНаСкладахОстатки.Качество,  
| ТоварыНаСкладахОстатки.КоличествоОстаток  
|ИЗ  
| РегистрНакопления.ТоварыНаСкладах.Остатки(  
| ,  
| Номенклатура = &Номенклатура  
| И Склад = &Склад) КАК ТоварыНаСкладахОстатки";

В этом случае требования соответствия индекса и запроса не нарушены. Данный запрос будет выполнен СУБД оптимальным способом. Следует обратить внимание на то, что порядок следования условий в запросе не обязан совпадать с порядком следования полей в индексе. Это не является проблемой и будет нормально обработано СУБД.

См. также

* [Индексы таблиц базы данных](https://its.1c.ua/db/metod8dev/content/1590/hdoc)
* [Эффективные условия запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100658.htm?_=1579516850)

### Разыменование ссылочных полей составного типа в языке запросов

#std654

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. В языке запросов возможно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении **ИЗ**, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся "через точку". Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей "через точку".

Например, в запросе

ВЫБРАТЬ  
ТоварныеЗапасы.Товар КАК Товар,  
ТоварныеЗапасы.Количество КАК Количество,  
ТоварныеЗапасы.Товар.Артикул КАК Артикул  
ИЗ  
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы  
...

кроме явно указанной в предложении **ИЗ** таблицы **РегистрНакопления.ТоварныеЗапасы** неявно участвует таблица **Справочник.Товары** для получения значения поля **Артикул**. А в случае использования ограничений доступа на уровне записей (RLS), к запросу добавляются ещё и таблицы, участвующие в RLS к таблице **Справочник.Товары**.

1.2. Большое число исходных таблиц запроса приводит к его усложнению и может значительно увеличивать время его выполнения. Особенно это важно помнить в тех случаях, когда поле таблицы ссылочного типа имеет [составной тип](https://its.1c.ua/db/content/v8std/src/100/300/i8100728.htm?_=1579516850) и может содержать ссылки на несколько таблиц. В таком случае, получение полей других таблиц "через точку" от такого поля составного типа приведет к соединениию со всеми таблицами, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Например, в приведенном ниже запросе получение даты регистратора приведет к неявному соединению с таблицами всех документов - регистраторов регистра **ТоварныеЗапасы**

ВЫБРАТЬ  
...  
ТоварныеЗапасы.Регистратор.Дата,  
...  
ИЗ  
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы  
...

Подобное получение данных "через точку" от ссылочных полей составного типа крайне нежелательно. Каждое исключение из этого правила должно тщательно анализироваться.

2.1. Следует избегать избыточности при создании полей составных ссылочных типов. Необходимо указывать ровно столько возможных типов для данного поля, сколько необходимо. Не следует без необходимости использовать типы "любая ссылка" или "ссылка на любой документ" и т.п.

Вместо этого следует более тщательно проанализировать прикладную логику и назначить для поля ровно те возможные типы ссылок, которые необходимы для решения задачи.

2.2. Для того чтобы избежать запросов с использованием большого числа исходных таблиц следует жертвовать компактностью хранения данных ради производительности и помещать соответствующие данные в исходную таблицу запроса.

Например, в регистре **ТоварныеЗапасы** можно завести реквизит **ДатаРегистратора**, заполнять его при проведении документов и использовать затем в запросах:

ВЫБРАТЬ  
...  
ТоварныеЗапасы.ДатаРегистратора,  
...  
ИЗ  
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы  
...

Это приведет к дублированию информации и некоторому (незначительному) увеличению ее объема, но может существенно повысить производительность и стабильность работы запроса.

2.3. При необходимости следует жертвовать компактностью и универсальностью кода ради производительности:

* Как правило, для выполнения конкретного запроса в данных условиях не нужны все возможные типы данной ссылки. В этом случае, следует ограничить количество возможных типов при помощи функции **ВЫРАЗИТЬ**.
* Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию **ВЫРАЗИТЬ** тот тип, который необходим при данных условиях.

Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.

Например, неправильно:

Запрос.Текст = "ВЫБРАТЬ  
| Продажи.Регистратор.Номер,  
| Продажи.Регистратор.Дата,  
| Продажи.Контрагент,  
| Продажи.Количество,  
| Продажи.Стоимость  
|ИЗ  
| РегистрНакопления.Продажи КАК Продажи  
|ГДЕ ...

В данном запросе используется обращение к реквизитам регистратора. Регистратор является полем составного типа, которое может принимать значения ссылки на один из 56 видов документов.    
SQL-текст этого запроса будет включать 56 левых соединений с таблицами документов. Это может привести к серьезным проблемам производительности при выполнении запроса.

Правильно:

Для решения данной конкретной задачи нет необходимости соединяться со всеми 56 видами документов. Условия запроса таковы, что при его выполнении будут выбраны только движения документов **РеализацияТоваровУслуг** и **ЗаказыПокупателя**. В этом случае можно значительно ускорить работу запроса, ограничив количество соединений при помощи функции **ВЫРАЗИТЬ()**.

Запрос.Текст = "ВЫБРАТЬ  
| ВЫБОР  
| КОГДА Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг  
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.РеализацияТоваровУслуг).Номер  
| КОГДА Продажи.Регистратор ССЫЛКА Документ.ЗаказПокупателя  
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.ЗаказПокупателя).Номер  
| КОНЕЦ ВЫБОРА КАК Номер,  
| ВЫБОР  
| КОГДА Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг  
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.РеализацияТоваровУслуг).Дата  
| КОГДА Продажи.Регистратор ССЫЛКА Документ.ЗаказПокупателя  
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.ЗаказПокупателя).Дата  
| КОНЕЦ ВЫБОРА КАК Дата,  
| Продажи.Контрагент,  
| Продажи.Количество,  
| Продажи.Стоимость  
|ИЗ  
| РегистрНакопления.Продажи КАК Продажи  
|ГДЕ  
| Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг  
| ИЛИ Продажи.Регистратор ССЫЛКА Документ.ЗаказыПокупателя";

Этот запрос является более громоздким и, возможно, менее универсальным (он не будет правильно работать для других ситуаций - когда возможны другие значения типов регистратора). Однако, при его выполнении будет сформирован SQL запрос, который будет содержать всего два соединения с таблицами документов. Такой запрос будет работать значительно быстрее и стабильнее, чем запрос в его первоначальном виде.

См. также

* [Самодостаточность регистров](https://its.1c.ua/db/content/v8std/src/100/300/i8100477.htm?_=1579516850)
* [Ограничения на использование реквизитов составного типа](https://its.1c.ua/db/content/v8std/src/100/300/i8100728.htm?_=1579516850)

### Ограничения на соединения с вложенными запросами и виртуальными таблицами

#std655

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. При написании запросов не следует использовать соединения с вложенными запросами. Следует соединять друг с другом только объекты метаданных или временные таблицы. Если запрос использует соединения с вложенными запросами, то его следует переписать с [использованием временных таблиц](https://its.1c.ua/db/content/v8std/src/200/200/i8100777.htm?_=1579516850) (не важно с какой стороны соединения находится вложенный запрос), кроме случая, когда вложенный запрос сканирует мало записей.

Если запрос содержит соединения с вложенными запросами, то это может привести к следующим негативным последствиям:

* Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков);
* Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других - очень медленно;
* Значительная разница по времени выполнения запроса на разных СУБД;
* Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время опять замедлиться.

Пример потенциально опасного запроса, использующего соединение с вложенным запросом:

ВЫБРАТЬ ...  
ИЗ Документ.РеализацияТоваровУслуг  
ЛЕВОЕ СОЕДИНЕНИЕ (  
   ВЫБРАТЬ ИЗ РегистрСведений.Лимиты  
   ГДЕ ...  
   СГРУППИРОВАТЬ ПО ...  
) ПО ...

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае, проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обоих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

1.2. Для вышеприведенного примера получится следующий пакетный запрос:

// Создать менеджер временных таблиц  
МенеджерВТ = Новый МенеджерВременныхТаблиц;  
Запрос = Новый Запрос;  
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;  
// Текст пакетного запроса  
Запрос.Текст = "  
  // Заполняем временную таблицу. Запрос к регистру лимитов.  
   | ВЫБРАТЬ ...  
   | ПОМЕСТИТЬ Лимиты  
   | ИЗ РегистрСведений.Лимиты  
   | ГДЕ ...  
   | СГРУППИРОВАТЬ ПО ...  
   | ИНДЕКСИРОВАТЬ ПО ...;  
   
  // Выполняем основной запрос с использованием временной таблицы  
   ВЫБРАТЬ ...  
   ИЗ Документ.РеализацияТоваровУслуг  
   ЛЕВОЕ СОЕДИНЕНИЕ Лимиты  
   ПО ...;"

Переписывание запроса по приведенной выше методике имеет своей целью упростить работу оптимизатору СУБД. В переписанном запросе все выборки, участвующие в соединениях будут представлять собой физические таблицы, и СУБД сможет легко определить размер каждой выборки. Это позволит СУБД гарантированно выбрать самый быстрый из всех возможных планов. Причем, СУБД будет делать правильный выбор независимо ни от каких условий. Переписанный подобным образом запрос будет работать одинаково хорошо на любых СУБД, что особенно важно при разработке тиражных решений. Кроме того, переписанный подобным образом запрос лучше читается, проще для понимания и отладки.

2. Если в запросе используется соединение с виртуальной таблицей языка запросов **1С:Предприятия** (например, **РегистрНакопления.Товары.Остатки**) и запрос работает с неудовлетворительной производительностью, то рекомендуется вынести обращение к виртуальной таблице в отдельный запрос с сохранением результатов во временной таблице (см. пункт 1.1).

3. Следует избегать неявных подзапросов, которые получаются при использовании вложенных соединений:

ВЫБРАТЬ ...  
ИЗ Справочник.Номенклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах  
        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций  
        ПО ...  
    ПО ...

Проблема в том, что, по сути, этот запрос аналогичен следующему:

ВЫБРАТЬ ...  
ИЗ Справочник.Номенклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ (  
        ВЫБРАТЬ ...  
        ИЗ РегистрНакопления.ТоварыНаСкладах  
            ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций  
            ПО ...)  
    ПО ...

Вместо вложенных соединений, как показано выше, следует использовать последовательные соединения:

ВЫБРАТЬ ...  
ИЗ Справочник.Номенклатура  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах  
    ПО ...  
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций  
    ПО ...

При этом следует понимать, что вложенные и последовательные соединения – это разные запросы, которые могут дать разный результат.

Если вложенное соединение использовано из предположения, что оно аналогично последовательному соединению, то следует просто переписать его на последовательное соединение.

Если вложенное соединение делается осмысленно, то от него следует отказаться, т.к. оно может существенно снизить производительность, как и соединение с подзапросом. Как и в случае с подзапросом, такое соединение можно заменить на соединение с временной таблицей, но лучше вначале подумать, как заменить его на последовательное соединение, т.к. оно будет работать эффективнее временной таблицы.

См. также

* [Использование вложенных запросов в условии соединения](https://its.1c.ua/db/content/v8std/src/200/200/i8100656.htm?_=1579516850)
* [Использование временных таблиц](https://its.1c.ua/db/content/v8std/src/200/200/i8100777.htm?_=1579516850)

### Ограничения на использование вложенных запросов в условии соединения

#std656

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Не следует использовать вложенные запросы в условии соединения. Это может привести к значительному замедлению запроса и (в отдельных случаях) к его полной неработоспособности на некоторых СУБД. Пример запроса с использованием вложенного запроса в условии соединения:

Запрос.Текст = "ВЫБРАТЬ  
   | ОстаткиТоваров.Номенклатура КАК Номенклатура,  
   | Цены.Цена КАК ЦенаПрошлогоМесяца  
   |ИЗ  
   | РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров  
   | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены  
   | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И  
   | Цены.Период В (  
   | ВЫБРАТЬ МАКСИМУМ(ЦеныПрошлогоМесяца.Период)  
   | ИЗ РегистрСведений.Цена КАК ЦеныПрошлогоМесяца  
   | ГДЕ ЦеныПрошлогоМесяца.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)  
   | И ЦеныПрошлогоМесяца.Номенклатура = ОстаткиТоваров.Номенклатура  
   | )  
   | ГДЕ ОстаткиТоваров.Склад = &Склад";

В данном случае вложенный запрос в условии соединения используется для получения как бы "среза последних" на конец предыдущего периода. Причем, для каждой номенклатуры период может быть разным.  
Подобный запрос рекомендуется переписать с использованием временных таблиц. Например, это можно сделать следующим образом:

Запрос.Текст = "  
  // Максимальные даты установки цен в прошлом периоде для данных номенклатур  
   |ВЫБРАТЬ  
   | ОстаткиТоваров.Номенклатура КАК Номенклатура,  
   | МАКСИМУМ(Цены.Период) КАК Период  
   |ПОМЕСТИТЬ ДатыПоНоменклатурам  
   |ИЗ  
   | РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров  
   | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены  
   | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И  
   | Цены.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)  
   | СГРУППИРОВАТЬ ПО ОстаткиТоваров.Номенклатура  
   | ГДЕ ОстаткиТоваров.Склад = &Склад;  
   
  // Выбрать данные по цене за найденный период  
   |ВЫБРАТЬ  
   | ДатыПоНоменклатурам.Номенклатура КАК Номенклатура,  
   | Цены.Цена КАК ЦенаПрошлогоМесяца  
   |ИЗ ДатыПоНоменклатурам  
   | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены  
   | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И  
   | Цены.Период = ДатыПоНоменклатурам.Период";

См. также

* [Ограничения на соединения с вложенными запросами и виртуальными таблицами](https://its.1c.ua/db/content/v8std/src/200/200/i8100655.htm?_=1579516850)
* [Использование временных таблиц](https://its.1c.ua/db/content/v8std/src/200/200/i8100777.htm?_=1579516850)

### Обращения к виртуальным таблицам

#std657

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При использовании виртуальных таблиц в запросах, следует передавать в параметры таблиц все условия, относящиеся к данной виртуальной таблице. Не рекомендуется обращаться к виртуальным таблицам при помощи условий в секции **ГДЕ** и т.п.

Такой запрос будет возвращать правильный (с точки зрения функциональности) результат, но СУБД будет намного сложнее выбрать оптимальный план для его выполнения. В некоторых случаях это может привести к ошибкам оптимизатора СУБД и значительному замедлению работы запроса.

Например, следующий запрос использует секцию **ГДЕ** запроса для выборки из виртуальной таблицы:

Запрос.Текст = "ВЫБРАТЬ  
| Номенклатура  
|ИЗ  
| РегистрНакопления.ТоварыНаСкладах.Остатки()  
|ГДЕ  
| Склад = &Склад";

Возможно, что в результате выполнения этого запроса сначала будут выбраны все записи виртуальной таблицы, а затем из них будет отобрана часть, соответствующая заданному условию.

Рекомендуется ограничивать количество выбираемых записей на самом раннем этапе обработки запроса. Для этого следует передать условия в параметры виртуальной таблицы.

Запрос.Текст = "ВЫБРАТЬ  
| Номенклатура  
|ИЗ  
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Склад = &Склад)";

2.1. При обращении к виртуальной таблице следует передавать в условия наиболее простые конструкции, например, "Измерение = Значение". Не рекомендуется использовать подзапросы и соединения(\*) в параметрах виртуальной таблицы, так как это приводит к медленной работе запроса.

\* Примечание: как явные соединения в подзапросах, так и неявные – при обращении к полям «через точку» от ссылки и соединения, добавляемые из ограничений доступа к данным (RLS), предусмотренных в ролях конфигурации.

2.2. При необходимости использовать подзапросы рекомендуется соблюдать следующие условия:

* в подзапросе только одна таблица, нет соединений с другими таблицами;
* если в подзапросе таблица табличной части (например, **Документ.Накладная.СписокТоваров**), то не должно быть обращения к реквизитам таблицы-шапки (**Накладная.Проведен**);
* если в подзапросе таблица, у которой могут быть табличные части (например, **Документ.Накладная**), то не должно быть обращений к табличным частям (например, **ГДЕ Документ.Накладная.СписокТоваров.Номенклатура = "1"**);
* если в подзапросе временная таблица, то не должно быть условий (раздела **ГДЕ**);
* если в подзапросе постоянная таблица, то условие (раздел **ГДЕ**) допустимо, только если условие выполняется для 80% (или более) случаев; отсутствие условия означает выполнение для 100% случаев.
* если в подзапросе постоянная таблица, то в ограничениях доступа к данным (RLS) не должно содержатся подзапросов и соединений (допускаются только простые условия вида **ГДЕ Реквизит = Значение**,**"ГДЕ Истина"**). Например, при использовании стандартных шаблонов RLS, входящих в состав подсистемы «Управление доступом» **Библиотеки стандартных подсистем** к запросу неявно добавляется конструкция Exists с несколькими подзапросами и соединениями. В таких случаях следует переписать исходный запрос с использованием временной таблицы или привилегированного режима.

Например, неправильно:

... ИЗ  
РегистрНакопления.ТоварыКОтгрузке.Остатки(  
&ДатаОтгрузки,  
&ОтображениеРаспоряжений  
 И ДокументОтгрузки.Склад = &Склад       -- неявное соединение «через точку»  
 ИЛИ ДокументОтгрузки В  
  (ВЫБРАТЬ                               -- подзапрос с соединением  
   Распоряжения.Распоряжение КАК ДокументОтгрузки  
  ИЗ  
  Документ.ЗаданиеНаПеревозку.Распоряжения КАК Распоряжения –- доступ к этому документу ограничен по сложному RLS, который неявно добавляет еще пару соединений  
  ВНУТРЕННЕЕ СОЕДИНЕНИЕ Документ.ЗаданиеНаПеревозку.СкладыПогрузки КАК  СкладыПогрузки  
  ПО  
  Распоряжения.Ссылка = СкладыПогрузки.Ссылка  
  И СкладыПогрузки.Склад = &Склад  
  И Распоряжения.Ссылка.Проведен         –- здесь и ниже обращения к реквизитам шапки  
  И Распоряжения.Ссылка.Статус В (...)))

правильно:

... ИЗ  
РегистрНакопления.ТоварыКОтгрузке.Остатки(  
 &ДатаОтгрузки,  
 Склад = &Склад                         -- теперь это реквизит регистра  
 ИЛИ ДокументОтгрузки В  
 (ВЫБРАТЬ  
  ЗаданияНаПеревозку.Распоряжение  
  ИЗ  
  ВременнаяТаблицаЗаданийНаПеревозку КАК ЗаданияНаПеревозку))  -- выборка из временной таблицы без условий

2.3. В случае, если нужно использовать несколько условий с подзапросами, следует выбрать одно, удовлетворяющее условиям выше и отфильтровывающее максимальное количество записей. Остальные условия следует накладывать на внешний запрос.

Кроме того, в ряде случаев можно обойтись и без перемещения условий на внешний запрос, если применять временные таблицы.  
Например, вместо условия (неправильно):

Номенклатура В (...) И Характеристика В (...) И Серия В(...)

правильно:

(Номенклатура, Характеристика, Серия) В (ВЫБРАТЬ Номенклатура, Характеристика, Серия ИЗ ВременнаяТаблицаТоваров)

См. также

* [Разыменование ссылочных полей составного типа в языке запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100654.htm?_=1579516850)
* [Использование параметра Условие при обращении к виртуальной таблице](http://its.1c.ru/db/metod81#content:5457:1) (статья на ИТС)
* [Эффективное обращение к виртуальной таблице «Остатки»](https://its.1c.ua/db/content/v8std/src/200/200/i8100733.htm?_=1579516850)

### Эффективные условия запросов

#std658

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Условия запросов должны быть написаны оптимально с точки зрения производительности, чтобы исключить существенное увеличение длительности выполнения запросов при увеличении объема данных в таблицах.

Поля основного условия в секциях ГДЕ, ПО и виртуальных таблицах должны быть проиндексированы. Основное условие может быть уточнено дополнительным условием, но объединять их следует только по И.

Важно понимать структуру индексов, которые получаются при индексировании полей и учитывать их при построении основного условия (см. [Индексы таблиц базы данных](https://its.1c.ua/db/metod8dev/content/1590/hdoc), [Несоответствие индексов и условий запроса](https://its.1c.ua/db/content/v8std/src/200/200/i8100652.htm?_=1579516850)). Например, при индексировании разных полей одного объекта метаданных создаются разные индексы, а не один в который помещаются все проиндексированные поля и использоваться основным условием будет только один из них.

**Основное условие** – это то, что позволяет ограничить объем выборки больше других условий и его составляющие объединены по И.

**Дополнительное условие** – это то, что объединено с основным условием по И и его составляющие могут быть любой сложности (НЕ, <>, +, -, /, \*, функции и т.п.).

Основное условие должно содержать только такие операции, которые позволяют выполнять поиск по индексу:

* для первого и всех используемых полей индекса, кроме последнего, только = и И;
* для последнего или единственного используемого поля индекса допустимо использовать =, >, <, >=, <=, ПОДОБНО, МЕЖДУ, В, ИЛИ (приводимое к В);
* нельзя использовать арифметические операции, функции, отрицания и неравенства.

Для условий в ГДЕ или в виртуальной таблице следует индексировать поля в основной таблице, из которой выполняется выборка.

Для условий в ПО ЛЕВОГО соединения следует индексировать поля в правой таблице.

Для условий в ПО ВНУТРЕННЕГО соединения следует индексировать поля в таблице с большим количеством записей.

Основное условие желательно строить таким образом, чтобы оно использовало индексы, которые автоматически создает платформа.

1.1. Описанные выше требования допустимо не соблюдать, если в таблицах, из которых выполняется выборка, или с которыми выполняется соединение, всегда будет мало данных (менее 1000 записей) или запросы с такими условиями выполняются очень редко.

1.2. Если записей в таблице много и выполнить указанные выше требования невозможно, то можно попробовать:

* преобразовать условия (см. п. 3, п. 4);
* добавить в таблицу заранее вычисляемые индексированные поля, которые заполняются при записи в нее и используются вместо сложного условия;
* если указанные выше рекомендации не помогли, то следует пересмотреть архитектуру решения так, чтобы можно было выполнить эти условия.

2. Оператор ИЛИ

2.1. В основном условии оператор ИЛИ можно использовать только для последнего из используемых или единственного поля индекса, когда оператор ИЛИ можно заменить на оператор В.

ПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле = &Значение1  
    ИЛИ Таблица.Поле = &Значение2

т.к. можно переписать при помощи оператора В (специально переписывать не нужно, можно оставить, как есть):

ГДЕ  
    Таблица.Поле В (&Значения)

НЕПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле1 = &Значение1  
    ИЛИ Таблица.Поле2 = &Значение2

нельзя переписать при помощи "В", но можно переписать при помощи "ОБЪЕДИНИТЬ ВСЕ" (каждое поле Поле1 и Поле2 должны быть проиндексированы):

ГДЕ  
    Таблица.Поле1 = &Значение1  
  
ОБЪЕДИНИТЬ ВСЕ  
  
ГДЕ  
    Таблица.Поле2 = &Значение1

Примечание: заменить ИЛИ на ОБЪЕДИНИТЬ ВСЕ можно не всегда, убедитесь, что результат будет действительно тем же, что и при ИЛИ, перед тем, как применять.

2.2. В дополнительном условии оператор ИЛИ можно использовать без ограничений.

ПРАВИЛЬНО 1:

ГДЕ  
    Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)  
    И // Дополнительное условие (можно использовать ИЛИ)  
    (Таблица.Поле2 = &Значение2 ИЛИ Таблица.Поле3 = &Значение3)

ПРАВИЛЬНО 2:

ГДЕ  
    (Таблица.Поле1 = &Значение1 ИЛИ Таблица.Поле1 = &Значение2)  
    И  
    (Таблица.Поле2 = &Значение3 ИЛИ Таблица.Поле2 = &Значение4)

т.к. можно переписать при помощи В (специально переписывать не нужно, можно оставить, как есть):

ГДЕ  
    Таблица.Поле1 В (&Значения1)   // Основное условие  
    И Таблица.Поле2 В (&Значения2) // Дополнительное условие (или наоборот)

3. Оператор ПОДОБНО

В основном условии для последнего из используемых или единственного поля индекса можно использовать оператор ПОДОБНО. Функции работы со строками, в некоторых случаях, можно привести к оператору ПОДОБНО и использовать его в основном условии.

НЕПРАВИЛЬНО 1:

ГДЕ  
    ПОДСТРОКА(Таблица.Поле, 1, 6) = "строка"

ПРАВИЛЬНО 1:

ГДЕ  
    Таблица.Поле ПОДОБНО "строка%"

НЕПРАВИЛЬНО 2:

ГДЕ  
    ПОДСТРОКА(Таблица.Поле, 3, 6) = "строка"

НЕПРАВИЛЬНО 2:

ГДЕ  
    Таблица.Поле ПОДОБНО "\_\_строка%" // Литерал не должен начинаться с символов "\_" или "%"

ПРАВИЛЬНО 2:

Добавить новое вычисляемое при записи в таблицу поле, которое будет содержать фрагмент ПОДСТРОКА(Таблица.Поле, 3, 6). Проиндексировать это поле и искать по следующему условию:

ГДЕ  
    Таблица.ВычисляемоеПоле ПОДОБНО "строка%"

4. Оператор МЕЖДУ

В основном условии для последнего из используемых или единственного поля индекса можно использовать оператор МЕЖДУ. Функции работы с датой, в некоторых случаях, можно привести к оператору МЕЖДУ и использовать его в основном условии.

НЕПРАВИЛЬНО:

ГДЕ  
    МЕСЯЦ(Таблица.Поле) = 1

ПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле МЕЖДУ &ДатаНачалаМесяца И &ДатаКонцаМесяца

Например, ДатаНачалаМесяца=01.01.2016, ДатаКонцаМесяца=31.01.2016 23:59:59

5. Выражение ВЫБОР

Выражение ВЫБОР можно использовать только в дополнительных условиях.

ПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)  
    И // Дополнительное условие (можно использовать ВЫБОР)  
    ВЫБОР  
        КОГДА Таблица.Поле2 = &Значение2  
            ТОГДА Таблица.Поле3 = &Значение3  
        ИНАЧЕ Таблица.Поле4 = &Значение4  
    КОНЕЦ

НЕПРАВИЛЬНО:

ГДЕ  
    ВЫБОР // Основное условие (поиск по индексу использоваться не будет)  
        КОГДА Таблица.Поле2 = &Значение2  
            ТОГДА Таблица.Поле3 = &Значение3  
        ИНАЧЕ Таблица.Поле4 = &Значение4  
    КОНЕЦ

6. Арифметические операции

Арифметические операции над полями можно выполнять только в дополнительных условиях.

ПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)  
    И // Дополнительное условие (можно выполнять арифметические операции)  
    Таблица.Поле2 - 1 > 0

НЕПРАВИЛЬНО:

ГДЕ  
    Таблица.Поле1 - 1 > 0 // Основное условие (поиск по индексу невозможен)

7. Если в конфигурации описано несколько ролей с разным ограничением доступа на уровне записей (RLS), то не следует назначать одному пользователю более одной такой роли. Если один пользователь будет включен, например, в две роли с RLS - бухгалтер и кадровик, то при выполнении всех его запросов к их условиям будут добавляться условия обоих RLS с использованием логического ИЛИ. Таким образом, даже если в исходном запросе нет условия ИЛИ, оно появится там после добавления условий RLS. Такой запрос так же может выполняться неоптимально - медленно и с избыточными блокировками.

Вместо этого следует:

* Пересмотреть состав ролей таким образом, чтобы к одному объекту метаданных давала доступ только одна роль (на чтение, запись и т.п.);
* При необходимости разработки нескольких ролей, предоставляющих доступ к одному объекту метаданных, задавать в них одинаковые условия RLS. В этом случае к тексту запроса будет добавлено только одно условие, без объединения по ИЛИ;
* Либо если это допустимо с точки зрения прикладной области, создать "смешанную" роль - "бухгалтер-кадровик" и прописать ее RLS таким образом, чтобы избежать использования ИЛИ в условии, а пользователя включить в эту одну роль.

См. также

* [Типичные причины неоптимальной работы запросов и методы оптимизации](https://its.1c.ru/db/metod8dev/content/5842/hdoc/_top/rls) (статья на ИТС)
* [Стандартные роли](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1579516850)
* [Настройка ролей и прав доступа](https://its.1c.ua/db/content/v8std/src/600/i8100689.htm?_=1579516850)

### Разрешение итогов для периодических регистров сведений

#std708 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Для периодических регистров сведений рекомендуется разрешить итоги, если выполнены все следующие условия:

* в регистре ожидается большой объем данных (например, оправданно для регистра с ценами номенклатуры; но не имеет смысла для регистра с курсами валют);
* в конфигурации предусмотрены частотные запросы к срезам последних на текущий момент времени и/или к срезам первых для получения актуальных данных (т.е. когда не задан период в параметрах виртуальных таблиц **СрезПервых**и **СрезПоследних**);
* при этом остальные условия для виртуальных таблиц **СрезПервых** и **СрезПоследних** задаются только на значения измерений (и разделителей, находящихся в режиме **Независимо и совместно**);
* в ограничениях доступа к данным регистра используются только измерения (и разделители, находящиеся в режиме **Независимо и совместно**).

Полный список всех условий, когда в запросах задействуются итоги регистра сведений, см. в [документации к платформе 1С:Предприятие](https://its.1c.ua/db/v83doc/content/63/hdoc/IssOgl3_5.14.1.1.%D0%9E%D0%B1%D1%89%D0%B0%D1%8F%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F%D0%BE%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%85%D1%81%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B9).

Например, если в конфигурации предусмотрены часто выполняющиеся запросы к регистру **ЦеныНоменклатуры** для получения текущих цен номенклатуры:

ВЫБРАТЬ  
 Номенклатура.Артикул КАК Артикул,  
 ЦеныНоменклатуры.Цена КАК Цена,  
 ...  
ИЗ  
 Справочник.Номенклатура КАК Номенклатура  
 ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.ЦеныНоменклатуры.СрезПоследних(, ВидЦены =  &ВидЦены) КАК ЦеныНоменклатуры  
ПО ЦеныНоменклатуры.Номенклатура = Номенклатура.Ссылка  
 ...

то при соблюдении всех остальных условий, перечисленных выше, установка свойства **Разрешить итоги: срез последних** существенно ускорит выполнение таких запросов, засчет того, что выборка будет выполняться напрямую из дополнительных таблиц, в которых хранятся только последние значения (для среза последних) и первые значения (для среза первых).

1.2. Кроме того, следует рассмотреть альтернативные варианты по пересмотру запросов к регистру таким образом, чтобы эти условия выполнялись.

Например, если в некоторых случаях данные в регистр **ЦеныНоменклатуры** записываются будущей датой, а при подборе товаров к этому регистру выполняется запрос всегда на текущую дату (дата явно задана в параметре виртуальной таблицы **СрезПоследних**), то итоги не будут ускорять выполнение таких запросов. Поскольку итоги строятся только для первых и последних записей регистра.

Однако если при открытии формы подбора товаров анализировать, есть ли регистраторы с будущей датой, и если их нет – выполнять другой запрос к срезу последних без установки даты, то такой запрос будет работать быстрее.

2. Во всех остальных случаях, не следует разрешать итоги для периодических регистров сведений. Прежде всего, если

* чаще всего (всегда) к виртуальным таблицам среза первых/последних периодического регистра сведений выполняются запросы на конкретный период (например, на дату документа).
* в условиях для виртуальных таблиц **СрезПервых** и **СрезПоследних** чаще всего (всегда) используются подзапросы и соединения (обращения «через точку» к полям связанных таблиц). Например, в этом случае:

ВЫБРАТЬ  
...  
ИЗ  
РегистрСведений.КурсыВалют.СрезПоследних(, Валюта.Код = &КодВалютыСклада) КАК КурсыВалют

3. Не требуется предусматривать в конфигурации отдельного механизма пересчета итогов, так как актуализация таблиц  итогов выполняется автоматически при каждой записи набора записей в регистр.

Исключение составляют отдельные случаи, когда актуализация итогов при записи отключалась принудительно с помощью вызова метода **РегистрСведенийМенеджер.УстановитьИспользованиеИтогов(Ложь)**.

### Эффективное обращение к виртуальной таблице «Остатки»

#std733

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Эффективность обращения к виртуальным таблицам во многом зависит от того, как построено обращение к этой таблице. Стандарт [Обращения к виртуальным таблицам](https://its.1c.ua/db/content/v8std/src/200/200/i8100657.htm?_=1579516850) описывает общие требования и рекомендации по работе с виртуальными таблицами. В этом стандарте изложены дополнительные рекомендации по повышению эффективности обращения к виртуальной таблице **Остатки** регистров накопления и бухгалтерии.

При обращении к любой виртуальной таблице платформа 1С:Предприятие генерирует запрос к СУБД, содержащий вложенный запрос. Самым эффективным вложенным запросом для чтения остатков будет чтение хранимой таблицы текущих остатков без применения группировки по измерениям. Платформа **1С:Предприятие** сгенерирует такой запрос, если будут соблюдены все перечисленные ниже условия:

* получение остатков ведется без указания даты;
* не используется разделение итогов (необходимо учитывать при использовании такого режима может снижаться параллельность записи в регистр. См. также [Режим разделения итогов для регистров накопления](https://its.1c.ua/db/content/v8std/src/200/400/i8100664.htm?_=1579516850), [Режим разделения итогов для регистров бухгалтерии](https://its.1c.ua/db/content/v8std/src/200/400/i8100663.htm?_=1579516850));
* внешний по отношению к виртуальной таблице запрос использует все измерения (в предложении ВЫБРАТЬ или в условиях соединения).

#### Пример.

Регистр накопления ОстаткиТовара содержит два измерения: Склад и Номенклатура, а также ресурс Количество. Необходимо запросом получить список всей номенклатуры, с указанием количества товаров на конкретном складе.

НЕПРАВИЛЬНО

ВЫБРАТЬ  
 СпрНоменклатура.Ссылка КАК Товар,  
 ЕСТЬNULL(ОстаткиТоваров.Остаток, 0 ) КАК Остаток  
ИЗ  
Справочник.Номенклатура КАК СпрНоменклатура  
 ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки(&СегодняшняяДата, Склад = &Склад) КАК ОстаткиТоваров  
ПО ОстаткиТоваров.Номенклатура = СпрНоменклатура.Ссылка

В этом запросе:

* в условия виртуальной таблицы передана дата, поэтому будет использована не только хранимые таблицы остатков, но и таблица движений. Т.к. необходимо получить текущие остатки, то дату в запрос передавать не нужно;
* измерение **Склад** не используется во внешнем по отношению к виртуальной таблице запросе, поэтому вложенный запрос остатков будет содержать группировку этому измерению.

ПРАВИЛЬНО

ВЫБРАТЬ  
 СпрНоменклатура.Ссылка КАК Товар,  
 ЕСТЬNULL(ОстаткиТоваров.Остаток, 0 ) КАК Остаток  
ИЗ  
Справочник.Номенклатура КАК СпрНоменклатура  
 ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки(, Склад = &Склад) КАК ОстаткиТоваров  
ПО ОстаткиТоваров.Номенклатура = СпрНоменклатура.Ссылка  
 И ОстаткиТоваров.Склад = &Склад

### Использование временных таблиц

#std777

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В общем случае, временные таблицы рекомендуется использовать для повышения производительности и стабильности выполнения запросов. Их можно использовать для других целей (например, для улучшения архитектуры кода), но при этом, нужно понимать, что это может в некоторых случаях приводить к снижению производительности.

2. Есть ситуации, когда временные таблицы не следует создавать или создание временных таблиц необходимо минимизировать.

2.1. Не следует создавать временные таблицы с очень большим объемом данных (сотни тысяч записей). Иначе это приведет к существенному снижению производительности при записи и исчерпанию свободного места на диске. Если алгоритму требуется работать с большим объемом данных, то он должен выполнять обработку этих данных порциями.

2.2. Следует максимально ограничивать количество данных, выбираемых во временную таблицу. Не следует помещать во временную таблицу больше данных, чем требуется последующим запросам.

2.3. Не следует помещать во временную таблицу поля, которые не используются в последующих запросах, т.к. время и место для их размещения тратится впустую.

2.4. Не следует создавать и удалять временные таблицы в цикле, если можно создать одну временную таблицу до выполнения цикла.

2.5. Не следует копировать одну временную таблицу в другую только ради того, чтобы переименовать первую таблицу во вторую. Вместо этого, следует передавать имя таблицы.

3. Временные таблицы следует всегда индексировать, когда это даст прирост производительности.

3.1. Индекс следует строить если:

3.1.1. Большая временная таблица участвует в соединении (не важно, с какой стороны). В индекс следует добавлять поля, участвующие в условии **ПО**.

3.1.2. Обращение к временной таблице выполняется в подзапросе конструкции логического оператора **В (...)**. В индекс следует добавлять поля временной таблицы из списка выбора, соответствующие перечисленным с левой стороны логического оператора **В (...)**.

См. также: стандарт 652 «[Несоответствие индексов и условий запроса](https://its.1c.ua/db/content/v8std/src/200/200/i8100652.htm?_=1579516850)».

3.2. Маленькие временные таблицы индексировать не нужно (менее 1000 записей).

3.3. Если условий выбора или соединений с временной таблицей больше одного, и только одно из них проверяется часто, то индекс следует строить для наиболее часто проверяемого условия.

См. также

* [Использование вложенных запросов в условии соединения](https://its.1c.ua/db/content/v8std/src/200/200/i8100656.htm?_=1579516850)
* [Ограничения на соединения с вложенными запросами и виртуальными таблицами](https://its.1c.ua/db/content/v8std/src/200/200/i8100655.htm?_=1579516850)

## Обработка и модификация данных

### Транзакции: правила использования

#std783

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Транзакции применяются для целостного изменения связанных данных, т.е. все действия с базой данных, выполняемые в рамках транзакции или выполняются целиком, или целиком откатываются.

1. Использование транзакций в **1С:Предприятии** обладает рядом особенностей:

* не поддерживаются вложенные транзакции (см. подробнее [Вложенность транзакций](https://its.1c.ua/db/metod8dev/content/2334/hdoc));
* при возникновении исключения в общем случае транзакция не может быть зафиксирована – при этом не важно, было ли это исключение обработано или нет (см. подробнее [Ошибки базы данных и транзакции](https://its.1c.ua/db/metod8dev/content/2313/hdoc), [Особенности работы объектов при отмене транзакции](https://its.1c.ua/db/metod8dev/content/2733/hdoc));
* транзакция может быть инициирована явно в прикладном коде при использовании метода **НачатьТранзакцию**. Так же платформа **1С:Предприятие** неявным образом начинает транзакцию при любой записи в базу данных (см. подробнее [Документация платформы. Механизм транзакций](https://its.1c.ua/db/v8313doc/bookmark/dev/TI000000527));

Эти особенности накладывают ряд требований к написанию кода с использованием транзакций. Несоблюдение этих требований может приводить к возникновению ошибок вида «В этой транзакции уже происходили ошибки», которые может быть крайне сложно воспроизвести и отладить.

1.1. Поскольку исключение не отменяет транзакцию сразу, но запрещает успешное завершение транзакции, то все вызовы **НачатьТранзакцию** с одной стороны и **ЗафиксироватьТранзакцию** или **ОтменитьТранзакцию** с другой стороны должны быть парными.

1.2. Начало транзакции и ее фиксация (отмена) должны происходить в контексте одного метода

Правильно

Процедура ЗаписатьДанныеВИБ()

    НачатьТранзакцию();

    Попытка  
        ... // чтение или запись данных  
        ДокументОбъект.Записать()  
        ЗафиксироватьТранзакцию();  
    Исключение  
        ОтменитьТранзакцию();  
        ... // дополнительные действия по обработке исключения  
    КонецПопытки;

КонецПроцедуры

Неправильно

Процедура ЗаписатьДанныеВИБ()  
   
    НачатьТранзакцию();  
    ЗаписатьДокумент();

КонецПроцедуры;

Процедура ЗаписатьДокумент()

    Попытка  
        ... // чтение или запись данных  
        ДокументОбъект.Записать()  
        ЗафиксироватьТранзакцию();  
    Исключение  
        ОтменитьТранзакцию();  
    ... // дополнительные действия по обработке исключения  
    КонецПопытки;

КонецПроцедуры

1.3. При использовании транзакций необходимо предусмотреть обработку исключений, придерживаясь следующих правил:

* метод **НачатьТранзакцию** должен быть за пределами блока **Попытка-Исключение** непосредственно перед оператором **Попытка**;
* все действия, выполняемые после вызова метода **НачатьТранзакцию**, должны находиться в одном блоке Попытка, в том числе чтение, блокировка и обработка данных;
* метод **ЗафиксироватьТранзакцию** должен идти последним в блоке **Попытка** перед оператором **Исключение**, чтобы  гарантировать, что после **ЗафиксироватьТранзакцию** не возникнет исключение;
* необходимо предусмотреть обработку исключений – в блоке **Исключение** нужно сначала вызвать метод **ОтменитьТранзакцию**, а затем выполнять другие действия, если они требуются;
* рекомендуется в блоке **Исключение** делать запись в [журнал регистрации](https://its.1c.ua/db/content/v8std/src/300/200/i8100498.htm?_=1580137113);
* при использовании вложенных транзакций (см. п. 1.4) в конце блока **Исключение** рекомендуется добавить оператор **ВызватьИсключение**.  В противном случае исключение не будет передано выше по стеку вызовов, там не сработает обработка исключения, внешняя транзакция не будет явным образом отменена и платформа вызовет исключение «В данной транзакции происходила ошибка»

Пример

НачатьТранзакцию();  
Попытка  
    БлокировкаДанных = Новый БлокировкаДанных;  
    ЭлементБлокировкиДанных = БлокировкаДанных.Добавить("Документ.ПриходнаяНакладная");  
    ЭлементБлокировкиДанных.УстановитьЗначение("Ссылка", СсылкаДляОбработки);  
    ЭлементБлокировкиДанных.Режим = РежимБлокировкиДанных.Исключительный;  
    БлокировкаДанных.Заблокировать();

    ... // чтение или запись данных  
  
    ДокументОбъект.Записать();

    ЗафиксироватьТранзакцию();  
Исключение  
    ОтменитьТранзакцию();

    ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),  
        УровеньЖурналаРегистрации.Ошибка,  
        ,  
        ,  
        ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));

    ВызватьИсключение; // есть внешняя транзакция

КонецПопытки;

1.4. Использование вложенных транзакций приводит к усложнению кода. Принимая решение об использовании этой возможности, нужно очень взвешенно оценить решаемую задачу: возможно, это усложнение просто не оправдано.

1.4.1. Не стоит усложнять код, явно используя метод **НачатьТранзакцию**, когда кроме записи объекта другие действия c базой данных не делаются – платформа при записи сама откроет транзакцию.

Не нужно явно открывать транзакцию тогда, когда не требуется выполнять [ответственное чтение данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100648.htm?_=1580137113). Например, обычно ответственное чтение не требуется при записи нового объекта (нового набора записей регистра).

При использовании методов **ПолучитьОбъект**(или **Прочитать**для наборов записей) необходимо анализировать должно ли чтение быть отвественным и в зависимости от этого принимать решение о явном использовании метода **НачатьТранзакцию**.

Правильно

Попытка  
    ДокументОбъект = Документы.ПриходнаяНакладная.СоздатьДокумент();  
    ... // действия по заполнению объекта  
    ДокументОбъект.Записать();  
Исключение  
    ... // действия по обработке исключения  
КонецПопытки;

Неправильно

НачатьТранзакцию();  
Попытка  
    ДокументОбъект = Документы.ПриходнаяНакладная.СоздатьДокумент();  
    ... // действия по заполнению объекта  
    ДокументОбъект.Записать();  
    ЗафиксироватьТранзакцию();  
Исключение  
    ОтменитьТранзакцию();  
КонецПопытки;

1.4.2. Если метод рассчитан на вызов только в рамках уже открытой транзакции (например, метод предназначен для вызова только из событий **ПередЗаписью**, **ОбработкаПроведения** и т.п.) в общем случае явным образом открывать в нем транзакцию не имеет никакого практического смысла.

1.4.3. При необходимости повысить качество сообщений об ошибках – на каждом уровне разработчик может предусмотреть свою обработку исключений, для чего, возможно, потребуется открыть вложенную транзакцию.

Пример

Вызывается метод ДобавитьЭлектроннуюПодпись. Внутри, если что-то пошло не так, нужно обработать исключение и добавить текст вида: «Не удалось добавить электронную подпись к объекту %ПредставлениеОбъекта% по причине:%ОписаниеОшибки%». В противном случае исключение будет обработано выше по стеку вызовов, например, при записи файла и будет выдано сообщение вида: «Не удалось записать файл %ИмяФайла% по причине: %ОписаниеОшибки%», где в «%ОписаниеОшибки%», будет просто указание на строчку кода и пользователю будет не понятно, зачем вообще программа записывала файл, если он просто его подписывал.

1.4.4. При обработке исключения, если транзакция все еще активна, например, исключение возникло во вложенной транзакции, нельзя обращаться к базе данных, так как это приведет к исключению «В этой транзакции уже происходили ошибки». При этом нужно учитывать, что обращение к базе данных может быть неявным, например, для получения представления ссылки.

2. Ограничение на длину транзакции.

2.1. В общем случае в рамках одной транзакции нужно выполнять только те действия, которые неделимы, исходя из бизнес-логики.

Пример

При проведении документа записывается документ и его движения в регистрах. Если не прошла запись хотя бы в один регистр вся операция проведения должна быть отменена.

2.1.1. Если с точки зрения бизнес-логики действия могут быть выполнены по отдельности, то их в общем случае не следует объединять в одну транзакцию.

2.1.2. Исключением из п.2.1.1 могут быть случаи, когда с целью оптимизации несколько несвязанных объектов обрабатываются в рамках одной транзакции. В этом случае необходимо взвешенно подходить к выбору порции обработки данных: нужно стремиться к достижению золотой середины между длительностью одной транзакции и объемом фиксируемых данных с одной стороны и количеством транзакций с другой.

2.2. Следует избегать транзакций, которые выполняются длительное время.

Пример

Неправильно

Для загрузки адресного классификатора ФИАС записывать все данные, относящиеся к одной версии классификатора в одной транзакции, для того, чтобы в случае ошибки откатить целиком загружаемую версию классификатора.

Т.к. данных по одной версии классификатора много (объем около 1 Гб), то для выполнения такой транзакции, во-первых, может не хватить оперативной памяти (особенно при использовании файловой информационной базы на 32-разрядной ОС), а, во-вторых, такая операция будет выполняться достаточно долго и ее нельзя будет оптимизировать за счет выполнения в несколько потоков.

Правильно

Разбить загрузку новой версии классификатора ФИАС на небольшие транзакции и реализовать функциональность по откату к предыдущей версии в случае ошибки.

См. также [Особенности использования транзакций при обмене данными](https://its.1c.ua/db/metod8dev/content/2274/hdoc)

2.2.1 Чем дольше выполняется транзакция, тем большее время будут заняты ресурсы сервера **1С:Предприятия** и СУБД. Как правило длинные транзакции занимают следующие ресурсы:

* в ходе выполнения транзакции все изменения в базе данных записываются в журнал транзакций, что необходимо для возможности откатить транзакцию;
* блокировки, установленные в транзакции, остаются до конца транзакции;
* на сервере **1С:Предприятия** блокировки занимают оперативную память;
* другие ресурсы, необходимые самой бизнес-логике, которая выполняется в транзакции.

Все это в целом может снижать эффективность использования ресурсов.

2.2.2. Если две транзакции пересекаются по блокируемым ресурсам, то транзакция, которая начала выполняться позже, будет ожидать возможность установления блокировки ограниченное время (по умолчанию – 20 секунд), после чего будет завершена с исключением «Превышено время ожидания установки блокировки». Поэтому длинные транзакции могут сильно снижать удобство параллельной работы пользователей.

Возникновение таких исключений – это повод провести анализ действий, которые выполняются в конфликтующих транзакциях

* возможно, какие-то действия можно вынести за транзакцию (см. п. 2.4);
* если действие вынести нельзя, то нужно постараться оптимизировать алгоритм его выполнения;
* так же нужно проанализировать оптимальность устанавливаемых блокировок (см. группу стандартов **Избыточные блокировки и методы оптимизации**)

2.3. В рамках транзакции нужно стремиться выполнять минимум действий – только те, которые нельзя в соответствии с бизнес-логикой выполнять вне транзакции. В частности:

* сложные, ресурсоемкие расчеты нужно стремиться делать до начала транзакции, если это позволяет бизнес-логика;
* если расчет должен выполняться в транзакции, то нужно стремиться сделать его как можно более простым. Например, контроль остатков можно делать уже после записи простым запросом к записываемому регистру;
* проверка заполнения объекта должна делаться вне транзакции (см. [Проверки, выполняемые в и вне транзакции записи объекта](https://its.1c.ua/db/content/v8std/src/100/400/i8100463.htm?_=1580137113));
* запросы, перед выполнением которых не нужно устанавливать блокировку данных, нужно стремиться выполнять до начала транзакции (см. [Ответственное чтение данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100648.htm?_=1580137113));
* запросы, выполняемые в рамках транзакций нужно стремиться оптимизировать (см. группу стандартов **Оптимизация запросов**)

### Использование управляемого режима блокировки

#std460

Область применения: управляемое приложение, обычное приложение.

В конфигурациях следует использовать "Управляемый" режим блокировок (свойство **Режим управления блокировкой данных** конфигурации устанавливается в значение **Управляемый**) и учитывать особенности работы в этом режиме, в частности:

* Чтение данных другими транзакциями будет невозможно только в том случае, если в текущей и других транзакциях устанавливаются несовместимые управляемые блокировки.
* Явная управляемая блокировка должна устанавливаться перед чтением данных:
* если считываются данные, которые в дальнейшем должны быть изменены;
* необходимо обеспечить неизменность считываемых данных до конца транзакции.
* При установке управляемой блокировки необходимо стремиться, чтобы блокировка была установлена только на те записи, которые будут обработаны системой в результате отработки программного кода.
* Не следует применять в запросах конструкцию **ДЛЯ ИЗМЕНЕНИЯ**, как не имеющую смысла в этом режиме.

При работе в автоматическом режиме управления блокировкой **1С:Предприятие** устанавливает высокую степень изоляции данных в транзакции на уровне СУБД. Это позволяет полностью исключить возможность получения нецелостных или некорректных данных без каких-либо специальных усилий со стороны прикладных разработчиков.

Но, при этом могут возникать избыточные блокировки на уровне СУБД. Эти блокировки связанны как с особенностями реализации механизмов блокировок в самой СУБД, так и с тем, что СУБД не может учитывать (и не учитывает) физический смысл и структуру объектов метаданных **1С:Предприятия**.

См. такке

* [Использование транзакций при чтении данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100648.htm?_=1580137113)
* [Общие сведения об избыточных блокировках](https://its.1c.ua/db/content/v8std/src/200/400/i8100659.htm?_=1580137113)
* методика перевода приложений в управляемый режим блокировки содержится в статье «[Блокировки данных в 1С:Предприятии 8](http://kb.1c.ru/articleView.jsp?id=30)»

### Блокировка данных объекта для редактирования из кода

#std490

Область применения: управляемое приложение, обычное приложение.

1.1. Прежде чем изменять существующий объект информационной базы из кода на встроенном языке, следует предварительно его заблокировать (установить «блокировку данных для редактирования» или «объектную блокировку»), тем самым, во-первых, убедиться, не заблокирован ли он другими объектами, во-вторых, попытаться предотвратить его изменение другими пользовательскими сеансами (или другими экземплярами объекта в этом же сеансе).

В противном случае, если при изменении и записи из встроенного языка не устанавливать блокировку объекта на время редактирования, то может возникнуть, например, ситуация, когда пользователь не сможет сохранить свои изменения, если эти же самые данные были конкурентно изменены в другом сеансе.

При этом блокировка данных для редактирования не запрещает запись заблокированных данных в других пользовательских сеансах (или в других экземплярах объекта в этом же сеансе), а лишь не позволяет нескольким объектам одновременно установить блокировку одних и тех же данных. В отличие от [транзакционных блокировок данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100648.htm?_=1580137113), пессимистическая блокировка данных для редактирования предназначена для обеспечения конкурентной работы пользователей с объектами информационной базы 1С:Предприятия (элементами справочников, документами и т.д.) Подробнее о блокировке данных для редактирования см. документацию по платформе **1С:Предприятие 8**.

1.2. Для блокировки данных для редактирования из встроенного языка следует вызывать метод объектов **Заблокировать** или метод глобального контекста **ЗаблокироватьДанныеДляРедактирования**.

Пример № 1. Требуется заблокировать объект и, если это удалось, модифицировать данные. В противном случае – проинформировать пользователя об отказе в выполнении операции с помощью сообщения вида:

«Не удалось заблокировать запись. Действие (изменение, удаление или блокировка записи) не выполнено. Ошибка блокировки объекта. Объект уже заблокирован: компьютер: <имя компьютера>, пользователь: <имя пользователя>, сеанс: <номер сеанса>, начат: <дата и время>, приложение: <тип клиентского приложения>».

ФайлОбъект = ДанныеФайла.Ссылка.ПолучитьОбъект();  
// Выполнить блокировку объекта от изменения другими режимами  
// или пользователями; в случае блокировки -  
// вывести пользователю сообщение об исключении.  
ФайлОбъект.Заблокировать();   
// Затем изменить и записать объект  
ФайлОбъект.Редактирует = Справочники.Пользователи.ПустаяСсылка();  
ФайлОбъект.Записать();

Аналогичным образом, можно воспользоваться методом глобального контекста **ЗаблокироватьДанныеДляРедактирования**:

ФайлОбъект = ДанныеФайла.Ссылка.ПолучитьОбъект();  
// Выполнить блокировку объекта от изменения другими режимами  
// или пользователями; в случае блокировки -  
// вывести пользователю сообщение об исключении.  
ЗаблокироватьДанныеДляРедактирования(ДанныеФайла.Ссылка);   
// Затем изменить и записать объект  
ФайлОбъект.Редактирует = Справочники.Пользователи.ПустаяСсылка();  
ФайлОбъект.Записать();

Пример № 2. Требуется пропустить обработку объекта, если он заблокирован для редактирования. При очередном вызове процедуры (например, из фонового или регламентного задания) будет предпринята повторная попытка изменения объекта.

Объект = ТекущаяВерсия.ПолучитьОбъект();  
// Выполнить блокировку объекта от изменения другими режимами  
// или пользователями  
УстановитьПолноеНаименование = Истина;  
Попытка  
  ФайлОбъект.Заблокировать();  
Исключение  
  // в случае блокировки - не выполнять изменение объекта  
  УстановитьПолноеНаименование = Ложь;  
  // записать предупреждение в журнал регистрации  
  ЗаписьЖурналаРегистрации(НСтр("ru = 'Фоновое обновление имен файлов'", Метаданные.ОсновнойЯзык.КодЯзыка),  
    УровеньЖурналаРегистрации.Предупреждение,, ФайлОбъект, ОписаниеОшибки());  
КонецПопытки;

// Пропустить обработку объекта, если он заблокирован.  
Если УстановитьПолноеНаименование Тогда  
  Объект.ПолноеНаименование = ПолноеНаименование;  
  Объект.Записать();  
КонецЕсли;

1.3. При редактировании данных в формах, платформа 1С:Предприятие автоматически устанавливает блокировку объекта, указанного в качестве основного реквизита формы.

2. Не следует проверять блокировку объектов для редактирования в следующих случаях:

* при выполнении отдельных операций, имеющих по логике работы больший приоритет по сравнению с интерактивными действиями пользователя. Например, загрузка данных при обмене;
* при действиях, которые гарантированно выполняются в монопольном режиме. Например, в процедурах обновления и первоначального заполнения данных информационной базы.

### Ответственное чтение данных

#std648

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Общие рекомендации по использованию транзакций при чтении данных

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  1.1. Если чтение данных из информационной базы должно быть ответственным, следует производить такое чтение в транзакции с предварительной установкой [управляемых блокировок](https://its.1c.ua/db/content/v8std/src/200/300/i8100460.htm?_=1580137113). В общем случае, ответственным следует считать любое чтение, на основе результатов которого производятся какие-либо изменения в информационной базе или принимаются решения. Например, ответственное чтение данных требуется в следующих случаях:   * Чтение данных при проведении, для последующего формирования движений; * Чтение данных для последующей целостной передачи в другую систему, например в программы типа «Клиент банк»; * Выполнение групповой обработки объектов, при реструктуризации данных в обработчиках отложенного и оперативного обновления ИБ (\*)  \* Примечание: перед модификацией ссылочных объектов, обычно, следует устанавливать на них [пессимистичные объектные блокировки](https://its.1c.ua/db/content/v8std/src/200/300/i8100490.htm?_=1580137113).   Неправильно:  // 1. Прочитать регистр сведений Запрос = Новый Запрос(   "ВЫБРАТЬ РАЗРЕШЕННЫЕ   | ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок   |ИЗ   | РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету   |ГДЕ   | ЗаметкиПоПредмету.Предмет = &Предмет"); Запрос.УстановитьПараметр("Предмет", ПредметЗаметок); Выборка = Запрос.Выполнить().Выбрать();   КоличествоЗаметок = 0; Если Выборка.Следующий() Тогда   КоличествоЗаметок = Выборка.КоличествоЗаметок; КонецЕсли;   // 2. Записать в регистр сведений НаборЗаписей = РегистрыСведений.ЗаметкиПоПредмету.СоздатьНаборЗаписей(); НаборЗаписей.Отбор.Предмет.Установить(ПредметЗаметок); НоваяЗапись = НаборЗаписей.Добавить(); НоваяЗапись.Предмет = ПредметЗаметок; НоваяЗапись.КоличествоЗаметок = КоличествоЗаметок + 1; НаборЗаписей.Записать();  Правильно:  // 1. Начать транзакцию для пакета из двух операций чтения и записи регистра НачатьТранзакцию();  Попытка   // 2. Установить исключительную блокировку на интересующий диапазон записей регистра,   // для того чтобы гарантировать, что в момент записи количество заметок не изменилось с момента чтения в каком-либо другом сеансе.   БлокировкаДанных = Новый БлокировкаДанных;   ЭлементБлокировкиДанных = БлокировкаДанных.Добавить("РегистрСведений.ЗаметкиПоПредмету");   ЭлементБлокировкиДанных.УстановитьЗначение("Предмет", ПредметЗаметок);   ЭлементБлокировкиДанных.Режим = РежимБлокировкиДанных.Исключительный;   БлокировкаДанных.Заблокировать();     // 3. Прочитать регистр сведений   Запрос = Новый Запрос(     "ВЫБРАТЬ РАЗРЕШЕННЫЕ     | ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок     |ИЗ     | РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету     |ГДЕ     | ЗаметкиПоПредмету.Предмет = &Предмет");   Запрос.УстановитьПараметр("Предмет", ПредметЗаметок);     Выборка = Запрос.Выполнить().Выбрать();     КоличествоЗаметок = 0;   Если Выборка.Следующий() Тогда     КоличествоЗаметок = Выборка.КоличествоЗаметок;   КонецЕсли;     // 4. Записать в регистр сведений   НаборЗаписей = РегистрыСведений.ЗаметкиПоПредмету.СоздатьНаборЗаписей();   НаборЗаписей.Отбор.Предмет.Установить(ПредметЗаметок);   НоваяЗапись = НаборЗаписей.Добавить();   НоваяЗапись.Предмет = ПредметЗаметок;   НоваяЗапись.КоличествоЗаметок = КоличествоЗаметок + 1;   НаборЗаписей.Записать();     ЗафиксироватьТранзакцию(); Исключение   // 5. Если при установке блокировки возникла исключительная ситуация из-за того, что регистр уже заблокирован в другом сеансе (или по другим причинам),   // отменить транзакцию и записать сведения об ошибке в журнал регистрации.   ОтменитьТранзакцию();   ЗаписьЖурналаРегистрации(НСтр("ru = 'Заметки'", ОбщегоНазначения.КодОсновногоЯзыка()), УровеньЖурналаРегистрации.Ошибка,,, ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));   ВызватьИсключение; КонецПопытки; |

В некоторых случаях, ответственное чтение не требуется в силу решаемой прикладной задачи, например:

* Получение данных динамическими списками;
* Поиск данных;
* Формирование большинства отчетов.

В некоторых случаях, ответственное чтение не требуется, так как конкурентная работа с данными маловероятна или полностью исключена, например:

* Обращение к условно постоянной информации. Например, чтение константы **ВалютаРегламентированногоУчета** или обращение к учетной политике;
* Действия, которые гарантированно выполняются в монопольном режиме. Например, в процедурах обновления и первоначального заполнения данных информационной базы;
* Действия над данными, доступ к которым имеет только один пользователь, поэтому конкурентная работа с ними маловероятна или полностью исключена.  
  Например, персональные данные, хранящиеся в «разрезе» пользователей;
* Мобильное приложение, где конкурентная работа с данными маловероятна или полностью исключена.

1.2. В большинстве случаев, при выполнении чтения в обработчиках событий связанных с модификацией данных, весь код обработчика выполняется в рамках системной транзакции, которая открыта платформой, и явно открывать новую транзакцию не требуется.

Например, в системной транзакции выполняются обработчики модулей объектов и соответствующие им подписки на события:

* [**ПередЗаписью**](https://its.1c.ua/db/content/v8std/src/100/400/i8100464.htm?_=1580137113);
* [**ПриЗаписи**](https://its.1c.ua/db/content/v8std/src/100/400/i8100465.htm?_=1580137113);
* [**ПередУдалением**](https://its.1c.ua/db/content/v8std/src/100/400/i8100752.htm?_=1580137113).

Подробнее – см. [документацию к платформе 1С:Предприятие](http://its.1c.ru/db/v8doc).

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  2. Выбор: исключительная или разделяемая блокировка  2.1. Если в транзакции производится ответственное чтение данных с их последующим изменением, необходимо установить **исключительную управляемую блокировку** (до выполнения чтения). В противном случае возможно возникновение взаимоблокировки.  Пример установки исключительной блокировки (без открытия транзакции – в предположении, что ранее уже была открыта системная транзакция):  // 1. Установить исключительную блокировку для ответственного чтения объекта с целью его дальнейшего изменения Блокировка = Новый БлокировкаДанных; ЭлементБлокировки = Блокировка.Добавить("Справочник.Приказы"); ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказСсылка); ЭлементБлокировки.Режим = РежимБлокировкиДанных.Исключительный; // можно не указывать, т.к. по умолчанию Исключительный Блокировка.Заблокировать();  // 2. Получить объект для его дальнейшей модификации Объект = ПриказСсылка.ПолучитьОбъект(); Если Объект = Неопределено Тогда // объект может быть уже удален в других сеансах   Возврат; КонецЕсли;  // 3. Выполнить блокировку объекта от изменения другими сеансами ЗаблокироватьДанныеДляРедактирования(ПриказСсылка);  // 4. Записать измененный объект Объект.Реквизит = ... Объект.Записать();  2.2. Если в транзакции производится ответственное чтение данных без их последующего изменения (например, для формирования движений), необходимо установить разделяемую блокировку на читаемые данные и исключительную блокировку на изменяемые данные.  Пример установки разделяемой блокировки (без открытия транзакции – в предположении, что ранее уже была открыта системная транзакция):  // 1. Установить разделяемую блокировку для ответственного чтения нескольких связанных объектов Блокировка = Новый БлокировкаДанных; ЭлементБлокировки = Блокировка.Добавить("Справочник.Приказы"); ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказСсылка); ЭлементБлокировки.Режим = РежимБлокировкиДанных.Разделяемый; Блокировка.Заблокировать();  // 2. Прочитать первый объект - приказ ПриказОбъект = ПриказСсылка.ПолучитьОбъект();  // 3. Прочитать второй объект – пользователя (автора приказа) Блокировка = Новый БлокировкаДанных; ЭлементБлокировки = Блокировка.Добавить("Справочник.Пользователи"); ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказОбъект.Автор); ЭлементБлокировки.Режим = РежимБлокировкиДанных.Разделяемый; Блокировка.Заблокировать();  АвторПриказа = ПриказОбъект.Автор.ПолучитьОбъект(); |

См. также

* [Общие сведения об избыточных блокировках](https://its.1c.ua/db/content/v8std/src/200/400/i8100659.htm?_=1580137113)
* [Транзакции: правила использования](https://its.1c.ua/db/content/v8std/src/200/300/i8100783.htm?_=1580137113)

### Чтение отдельных реквизитов объекта из базы данных

#std496

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

При чтении отдельных реквизитов объекта из базы данных следует иметь в виду, что вызов метода **ПолучитьОбъект** или обращение к реквизитам объекта через точку от ссылки приводит к загрузке объекта из базы целиком, вместе с его табличными частями.

Поэтому для чтения значений отдельных реквизитов из базы данных следует использовать запрос. Например, неправильно:

Процедура ЗаполнитьКодИНаименованиеСтраны()  
   
 СтранаСсылка = … // получаем ссылку на элемент справочника   
 КодСтраны = СтранаСсылка.Код; // первое обращение загружает объект целиком  
 НаименованиеСтраны = СтранаСсылка.Наименование;  
   
КонецПроцедуры

правильно:

Процедура ЗаполнитьКодИНаименованиеСтраны()  
   
 Запрос = Новый Запрос(  
  "ВЫБРАТЬ  
  | СтраныМира.Код,  
  | СтраныМира.Наименование  
  |ИЗ  
  | Справочник.СтраныМира КАК СтраныМира  
  |ГДЕ  
  | СтраныМира.Ссылка = &Ссылка");  
 Запрос.УстановитьПараметр("Ссылка", Ссылка);  
   
 Выборка = Запрос.Выполнить().Выбрать();  
 Выборка.Следующий();

 КодСтраны = Выборка.Код;  
 НаименованиеСтраны = Выборка.Наименование;

КонецПроцедуры

Для упрощения синтаксиса рекомендуется также использовать специальные функции **ЗначенияРеквизитовОбъекта** или **ЗначениеРеквизитаОбъекта** (входят в состав [Библиотеки стандартных подсистем](http://v8.1c.ru/libraries/ssl/index.htm)).  
В этом случае исходный пример будет выглядеть так:

Процедура ЗаполнитьКодИНаименованиеСтраны()

 ЗначенияРеквизитов = ОбщегоНазначения.ЗначенияРеквизитовОбъекта(СтранаСсылка, "Код, Наименование");  
 КодСтраны = ЗначенияРеквизитов.Код;  
 НаименованиеСтраны = ЗначенияРеквизитов.Наименование;  
   
КонецПроцедуры

### Запись событий в историю работы пользователя

#std497

Область применения: управляемое приложение.

В историю работы автоматически попадают события интерактивного добавления или изменения объектов информационной базы (документов, элементов справочника и пр.) Дополнительно рекомендуется записывать в историю работы пользователя и другие события, которые приводят к записи объектов в результате действий пользователя. Например: команда «Поместить файл» для элемента справочника **Файлы**.

Для добавления событий в историю работы пользователя предназначен объект **ИсторияРаботыПользователя** типа **МенеджерИсторииРаботыПользователя**. Пример:

&НаКлиенте  
Процедура ПоместитьФайл(Команда)  
  // Поместить файл в базу  
  // …  
  // И добавить событие в историю работы пользователя  
  ИсторияРаботыПользователя.Добавить(ПолучитьНавигационнуюСсылку(Объект.Ссылка));  
КонецПроцедуры

## Избыточные блокировки и методы оптимизации

### Общие сведения об избыточных блокировках

#std659

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1. Проектные ошибки при выборе и проектировании структуры того или иного объекта метаданных для реализации прикладной функциональности, могут привести к большому количеству избыточных блокировок и как следствие к серьезному падению общей производительности системы.  Ожидание на блокировке данных происходит в том случае, если две различные сессии **1С:Предприятия** пытаются захватить один и тот же ресурс. При работе с разными ресурсами ожидание на блокировке не происходит. В данном контексте термин «ресурс» используется в качестве обозначения неделимой совокупности данных, которая блокируется (или не блокируется) только вся целиком.  Таким образом, вопрос сводится к тому, какие именно ресурсы захватываются при выполнении того или иного действия с данными. Или, иначе говоря, насколько «мелко нарезаны» данные **1С:Предприятия**.  При анализе структуры метаданных следует обратить внимание на следующие объекты:   * [Последовательность](https://its.1c.ua/db/content/v8std/src/200/400/i8100662.htm?_=1580137113) * [Регистры бухгалтерии](https://its.1c.ua/db/content/v8std/src/200/400/i8100663.htm?_=1580137113) * [Регистры накопления](https://its.1c.ua/db/content/v8std/src/200/400/i8100664.htm?_=1580137113)   См. также   * [Блокирующее чтение остатков в начале транзакции](https://its.1c.ua/db/content/v8std/src/200/400/i8100661.htm?_=1580137113) * [Использование управляемого режима блокировки](https://its.1c.ua/db/content/v8std/src/200/400/i8100460.htm?_=1580137113) * [Несоответствие индексов и условий запроса](https://its.1c.ua/db/content/v8std/src/200/200/i8100652.htm?_=1580137113) * [Разыменование ссылочных полей составного типа в языке запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100654.htm?_=1580137113) * [Запросы, выполняющие соединение с вложенными запросами или виртуальными таблицами](https://its.1c.ua/db/content/v8std/src/200/200/i8100655.htm?_=1580137113) * [Использование вложенных запросов в условии соединения](https://its.1c.ua/db/content/v8std/src/200/200/i8100656.htm?_=1580137113) * [Фильтрация виртуальных таблиц без использования параметров](https://its.1c.ua/db/content/v8std/src/200/200/i8100657.htm?_=1580137113) * [Ограничение на использование логического ИЛИ в условиях запросов](https://its.1c.ua/db/content/v8std/src/200/200/i8100658.htm?_=1580137113) |

### Сдвиг границы последовательности документов

#std662

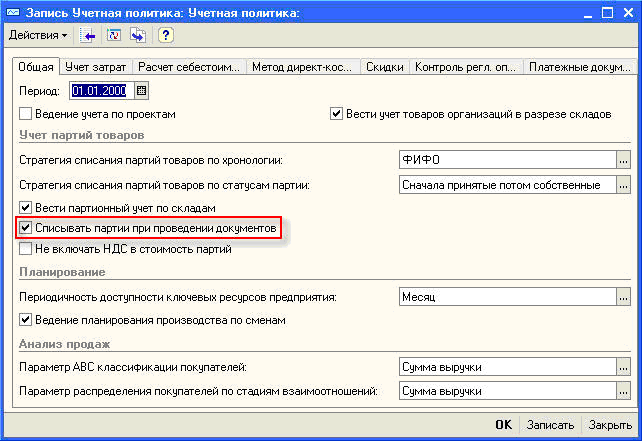
Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Не рекомендуется двигать границу последовательности при проведении документов. Это может привести к возникновению ожиданий на блокировках и снижению общей производительности системы. Операцию движения границы последовательности следует вынести из оперативных операций в регламентные, например выполнять регламентной обработкой с заданной частотой.

Граница последовательности по одному набору значений измерений является одним ресурсом. Это означает, что при движении границы последовательности по одному набору значений измерений разные пользователи будут пытаться захватить один и тот же ресурс, то есть будут блокировать друг друга.

Пример

В системе, построенной на базе УПП версий 1.х, используется учетная политика, предполагающая вычисление себестоимости списываемых товаров в оперативном режиме (непосредственно в момент списания).

Алгоритмы партионного учета в УПП используют последовательность «Партионный учет», имеющую одно измерение: **Организация**.

Для вычисления себестоимости списываемых товаров при проведении расходного документа необходимо переместить границу последовательности для данной организации на момент времени проводимого документа.

Если два пользователя будут одновременно проводить расходные документы по одной организации (что весьма вероятно), то они будут блокировать друг друга. Такое поведение системы не является особенностью реализации последовательностей в **1С:Предприятии**, но продиктовано требованиями самого алгоритма – необходимо знать точную последовательность расположения документов. То есть все конкурирующие по времени (одновременно проводящиеся) документы должны выстроиться друг за другом.

Движение границы последовательности при оперативном проведении документов способно значительно снизить общую производительности системы. В данном случае правильным решением было бы выключение этого флажка, то есть отказ от оперативного расчета себестоимости при проведении документов. Вместо этого следует использовать регламентную обработку, входящую в состав УПП, которая будет вычислять себестоимость с некоторой заданной частотой.

См. также

* «[Общие сведения об избыточных блокировках](https://its.1c.ua/db/content/v8std/src/200/400/i8100659.htm?_=1580137113)»

### Режим разделения итогов для регистров бухгалтерии

#std663

Область применения: управляемое приложение, обычное приложение.

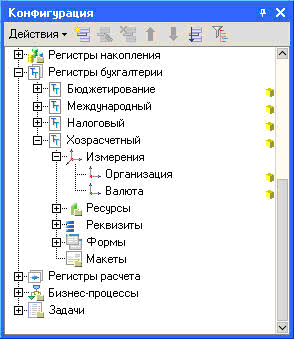
1. Если в системе осуществляется оперативная запись движений по бухгалтерскому регистру в многопользовательском режиме, то рекомендуется включить для данного регистра режим разделения итогов. При включенном режиме разделения итогов пользователи смогут параллельно обновлять таблицу остатков даже в том случае, если у них совпадает период, счет и значения измерений.

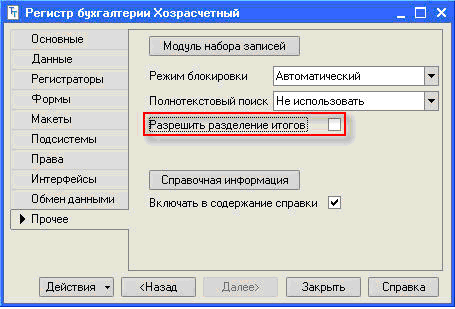
В противном случае таблица остатков регистра бухгалтерии может стать узким местом при конкурентной работе большого количества пользователей.

См. также [Блокирующее чтение остатков в начале транзакции](https://its.1c.ua/db/content/v8std/src/200/400/i8100661.htm?_=1580137113)

#### Пример 1

В конфигурации определен регистр **Хозрасчетный** с измерениями **Организация**и **Валюта**.

  
При этом запрещено разделение итогов регистра:

   
Предположим, что два пользователя одновременно проводят документы, которые осуществляют движение по данному регистру.  
Пользователи будут блокировать друг друга в том случае, если движения:

* относятся к одному и тому же периоду;
* относятся к одному и тому же счету;
* имеют одинаковые значения измерений, то есть организацию и валюту.

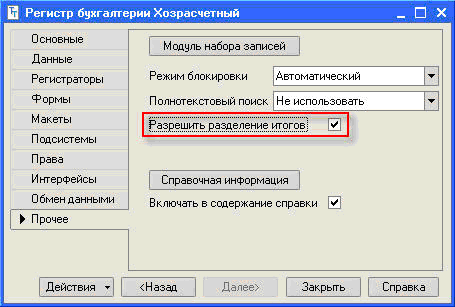
В реальной жизни одновременное выполнение перечисленных условий является весьма вероятным, поскольку большинство пользователей будет работать в одном периоде, с одним счетом и с одинаковыми значениями измерений (**организация** и **валюта**). Это может привести к возникновению ожиданий на блокировках и снижению общей производительности системы.

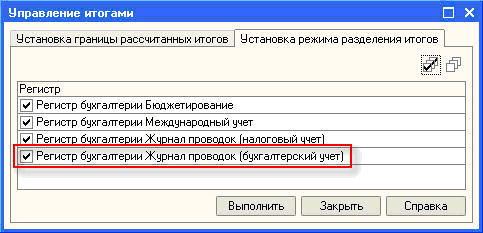
Для решения этой проблемы следует включить режим разделения итогов (см. следующий пример).

#### Пример 2

Если для этого же регистра разрешить и включить режим разделения итогов, то ситуация изменится.

Разрешим режим разделения итогов (в режиме **конфигурирования**):

    
Включим режим разделения итогов (в режиме **1С:Предприятия**):



После этого конкурирующие пользователи смогут параллельно записывать движения по регистру даже в том случае, если совпадают период, номер счета и значения всех измерений. Однако, если при этом осуществляется [контроль остатков по данному регистру](https://its.1c.ua/db/content/v8std/src/200/400/i8100661.htm?_=1580137113), то эффекта от включения режима разделения не будет.

См. также

* [Устройство и использование режима разделения итогов регистров](http://its.1c.ru/db/metod81#content:1393:1) (статья на ИТС)
* [Эффективное обращение к виртуальной таблице «Остатки»](https://its.1c.ua/db/content/v8std/src/200/200/i8100733.htm?_=1580137113)

### Режим разделения итогов для регистров накопления

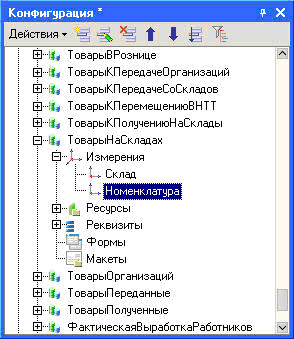
#std664

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При проектировании регистра накопления следует помнить, что остатки по одному набору измерений хранятся в одном ресурсе регистра. То есть, степень параллельности при работе с регистром фактически зависит от состава его измерений. Состав измерений необходимо подбирать в соответствии с тем, насколько мелко должны быть «нарезаны» остатки, исходя из прикладной функциональности системы.

2. В том случае, если состав измерений не позволяет обеспечить необходимую параллельность при работе с регистром, рекомендуется использовать режим разделения итогов аналогично [регистрам бухгалтерии](https://its.1c.ua/db/content/v8std/src/200/400/i8100663.htm?_=1580137113).  
Следует учитывать, что режим разделения итогов не обеспечит параллельность при контроле остатков по регистру. Если контроль остатков необходим, то следует перенести его [как можно ближе к концу транзакции](https://its.1c.ua/db/content/v8std/src/200/400/i8100661.htm?_=1580137113).

Рассмотрим в качестве примера регистр накопления **ТоварыНаСкладах** со следующим составом измерений:



#### Пример 1

Предположим два пользователя одновременно проводят документы, которые записывают движения в данный регистр накопления. При этом первый пользователь пишет следующий набор записей:

|  |  |  |
| --- | --- | --- |
| № | Склад | Номенклатура |
| 1 | Основной склад | Кресло-качалка |
| 2 | Основной склад | Кухонный гарнитур "Тинга-2" |
| 3 | Склад №2 | Мебельный гарнитур "Торэ" |

Второй пользователь пишет в этот же регистр следующий набор записей:

|  |  |  |
| --- | --- | --- |
| № | Склад | Номенклатура |
| 1 | Основной склад | Мебельный гарнитур "Торэ" |
| 2 | Склад №2 | Кресло-качалка |

Эти наборы записей не содержат строк, совпадающих по значениям всех измерений, поэтому ожидание на блокировке в данном случае не возникнет.

#### Пример 2

Предположим, что второй пользователь записывает следующий набор записей

|  |  |  |
| --- | --- | --- |
| № | Склад | Номенклатура |
| 1 | Основной склад | Кухонный гарнитур "Тинга-2" |
| 2 | Склад №2 | Кресло-качалка |
| 3 | Оптовый склад | Спальный гарнитур "Инга-М" |

Этот набор записей содержит строку (№1), совпадающую по значениям всех измерений со строкой набора записей первого пользователя (№2).

В этом случае возникнет ожидание на блокировке и один из пользователей будет дожидаться окончания операции другого пользователя, то есть общая производительность системы снизится.

Регистр накопления также поддерживает режим разделения итогов, который позволит в данной ситуации избежать блокировки (см. Пример 3).

#### Пример 3

Если для данного регистра накопления включен режим разделения итогов, то будет возможна параллельная запись двух наборов, даже в том случае если они содержат одинаковые (по значениям измерений) строки.

Однако, если при этом используется контроль остатков, то режим разделения итогов не даст положительного эффекта. Иначе говоря, режим разделения итогов для регистра накопления работает аналогично [регистру бухгалтерии](https://its.1c.ua/db/content/v8std/src/200/400/i8100663.htm?_=1580137113). Для снижения влияния этой блокировки на общую производительность системы [нужно перенести контроль остатков как можно ближе к концу транзакции](https://its.1c.ua/db/content/v8std/src/200/400/i8100661.htm?_=1580137113).

См. также

* [Устройство и использование режима разделения итогов регистров](http://its.1c.ru/db/metod81#content:1393:1) (статья на ИТС)
* [Эффективное обращение к виртуальной таблице «Остатки»](https://its.1c.ua/db/content/v8std/src/200/200/i8100733.htm?_=1580137113)

### Блокирующее чтение остатков в начале транзакции

#std661

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. В ряде случаев необходимо выполнять блокирующее чтение итогов. Примером такой задачи является контроль остатков при проведении документа. Если в результате проведения документа остатки станут отрицательными, то транзакция должна быть отменена (проводить такой документ нельзя).

1.2. Операция чтения остатков должна быть блокирующей, то есть необходимо запретить двум пользователям одновременно читать один и тот же остаток за период, счет и значение измерения.

Если чтение будет неблокирующим, то возможна ситуация, при которой два пользователя одновременно прочитают один и тот же остаток (например 10 единиц) и примут решение о возможности списания части этого остатка. Если сумма списаний двух пользователей будет больше 10, то в итоге остаток получится отрицательным.

Например, первый пользователь спишет 8 единиц (8 меньше 10, следовательно операция разрешена), а второй пользователь спишет 6 единиц (на таком же основании). Результатом будет -4 единицы остатка, что недопустимо с точки зрения прикладной логики системы.

2. Обычно, для контроля остатков используется запрос в модуле набора записей регистра, который идет перед записью набора. При этом возможны следующие проблемы:

* Разработчик, как правило, не контролирует порядок записи движений в разные регистры - запись обычно осуществляется автоматически платформой **1С:Предприятия**. Запрос контроля остатков реализуется в модуле набора записей и вызывается при записи движений регистра. Если этот регистр будет записываться в начале транзакции (например, первым), то установленная блокировка будет мешать работе других пользователей в течение длительного периода времени (пока будут записываться все остальные регистры), и ее влияние на производительность системы может оказаться неоправданно большим.
* В некоторых случаях, возможно, нет необходимости в контроле остатков, поскольку записываемые движения заведомо не могут привести к получению отрицательных остатков.

Для того чтобы минимизировать влияние блокирующего чтения остатков на производительность системы, необходимо:

* Проанализировать, какие именно остатки нуждаются в блокирующем чтении и при каких обстоятельствах. Например, контроль остатков не требуется при проведении приходного документа, поскольку он может только увеличить остатки. Так же не требуется контролировать остатки при перепроведении документа, который списывает в этот раз не больше остатков чем при первом проведении (этот контроль уже проводился). И так далее.
* В начале транзакции (например, в обработчике **ОбработкаПроведения** документа) в явном виде записать движения по всем регистрам, которые в данном случае не требуют контроля остатков. Следует всегда придерживаться одинакового порядка записи регистров (например, алфавитного). Необходимо обратить внимание на то, что у всех записываемых регистров накопления и бухгалтерии должен быть включен разделитель итогов, а у наборов записей свойство **БлокироватьДляИзменения** должна быть установлена в значение Ложь.
* Выполнить все остальные действия, которые должны быть выполнены в рамках этой транзакции.
* В самом конце транзакции в явном виде записать движения по тем регистрам, которые требуют контроля остатков. Для наборов записей этих регистров следует установить опцию **БлокироватьДляИзменения** в значение Истина. Это необходимо для предотвращения взаимоблокировки. В этом случае при записи набора записей будет установлена блокировка остатков регистра по данному набору значений измерений.
* Для каждого регистра выполнить запрос контроля остатков. Следует обратить внимание, что в данном случае нет необходимости использовать явную управляемую блокировку (опцию **ДЛЯ ИЗМЕНЕНИЯ** - в автоматическом режиме), поскольку проверяемые остатки уже заблокированы их записью на предыдущем шаге. Запрос должен считывать только отрицательные остатки по заданному набору значений измерений. Если такие записи имеются, то транзакция должна быть отменена. Если запрос вернул пустой результат, то транзакция должна быть зафиксирована.

#### Пример

В процедуре **ПередЗаписью** модуля набора записей регистра бухгалтерии **Хозрасчетный** выполняется следующий запрос:

Запрос.Текст = "ВЫБРАТЬ  
| СуммаОстаток,  
| СуммаОстатокДт,  
| СуммаОстатокКт  
|ИЗ  
| РегистрБухгалтерии.Хозрасчетный.Остатки(&Период, &Счет, , Организация = &Организация)";

При выполнении этого запроса будут прочитаны (и заблокированы от записи) остатки по указанному условию для всех пользовательских подключений. То есть, разные ресурсы (созданные режимом разделения итогов) будут как бы объединены в один. По этой причине параллельность останется такой же, как если бы режим разделения итогов не был включен.

Для того чтобы минимизировать влияние этой блокировки на общую производительность системы, рекомендуется перенести ее как можно ближе к концу транзакции. Например, можно вынести эту проверку в модуль документа в обработчик события **ПриПроведении** после записи (в явном виде) всех движений по всем регистрам.

См. также

* [Использование транзакций при чтении данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100648.htm?_=1580137113)
* [Регистры бухгалтерии](https://its.1c.ua/db/content/v8std/src/200/400/i8100663.htm?_=1580137113)

# [Соглашения при написании кода](https://its.1c.ua/db/v8std/browse/13/-1/31)

## Оформление модулей

### Тексты модулей

#std456

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Тексты модулей должны быть написаны на русском языке.

Исключение составляют [веб-сервисы](https://its.1c.ua/db/content/v8std/src/100/100/i8100550.htm?_=1580137113), имена методов и параметров которых рекомендуется задавать на английском языке (например, метод **TestConnection** веб-сервиса **EnterpriseDataExchange**), а также различные идентификаторы сторонних информационных систем, например, названия полей «внешних» структур данных, которые программно обрабатываются в коде.

1.1. В текстах модулях не допускается использовать букву "ё".

Исключения составляют интерфейсные тексты, выводимые пользователю в сообщениях, формах и справке, где употребление буквы «ё» в ряде случаев допустимо. Подробнее см. [Тексты](https://its.1c.ua/db/content/v8std/src/1%C2%A0200/1%C2%A0800/i8100598.htm?_=1580137113).

1.2. Кроме того, в текстах модулей не допускается использовать неразрывные пробелы и знак минус "-" в других кодировках (короткое, длинное тире и т.п.).

Такие символы часто оказываются в тексте модулей при копировании из офисных документов и приводят к ряду сложностей при разработке. Например, не работает поиск фрагментов текста, включающих «неправильные» минусы и пробелы; некорректно выводятся подсказки типов параметров процедур и функций в конфигураторе и расширенная проверка в 1С:EDT; указание «неправильного» минуса в выражениях приведет к синтаксической ошибке.

|  |
| --- |
| Методическая рекомендация (полезный совет)  2. Программные модули не должны иметь неиспользуемых процедур и функций. |

3. Программные модули не должны иметь закомментированных фрагментов кода, а также фрагментов, которые каким-либо образом связаны с процессом разработки (отладочный код, служебные отметки, например, TODO, MRG и т.п.) и с конкретными разработчиками этого кода. Например, недопустимо оставлять подобные фрагменты в коде после завершения отладки или рефакторинга:

Процедура ПередУдалением(Отказ)

// Если Истина Тогда

// Сообщение("Для отладки");

// КонецЕсли;

КонецПроцедуры

также неправильно:

Процедура ПередУдалением(Отказ)

Если Истина Тогда

// Иванов: доделать

КонецЕсли;

КонецПроцедуры

Правильно: после завершения отладки или рефакторинга удалить обработчик **ПередУдалением** из кода.

4. Тексты модулей оформляются по принципу "один оператор в одной строке". Наличие нескольких операторов допускается только для "однотипных" операторов присваивания, например:

НачальныйИндекс = 0; Индекс = 0; Результат = 0;

5. Текст модуля должен быть оформлен синтаксическим отступом. Для синтаксического отступа используется табуляция (а не пробелы, чтобы при смене числа знаков в табуляции выравнивание текста сохранялось).

Размер табуляции - стандартный (4 символа).  
  
5.1. С крайней левой позиции должны начинаться только:

* операторы **Процедура**, **КонецПроцедуры**, **Функция**, **КонецФункции**;
* операторы предварительного объявления процедур и функций;
* заголовки (описания) процедур и функций;
* объявление переменных модуля;
* операторы "раздела основной программы" (с учетом синтаксического отступа);
* директивы компилятора **&НаКлиенте**, **&НаСервере**и т.д.
* инструкции препроцессора (в т.ч. **#Область** и **#КонецОбласти**)

5.2. Процедуры **НачатьТранзакцию** и **ЗафиксироватьТранзакцию** не являются операторными скобками, поэтому текст внутри этих процедур не сдвигается.

6. При длине строки более 120 символов следует использовать переносы. Строки длиннее 120 символов делать не рекомендуется, за исключением тех случаев, когда перенос невозможен (например, в коде определена длинная строковая константа, которая выводится без переносов в окно сообщений с помощью объекта **СообщениеПользователю**).

См. также: [*Перенос выражений.*](https://its.1c.ua/db/content/v8std/src/300/200/i8100444.htm?_=1580137113)

7.1. Тексты модулей могут содержать **комментарии**. Комментарии должны быть достаточно понятными, чтобы пояснять работу модуля или комментируемого оператора. Тексты комментариев должны составляться по правилам русского языка, в деловом стиле, быть эмоционально сдержанными и не содержать слов, не относящихся к функциональности программы.

7.2. Небольшие комментарии пишутся в конце строки, которую комментируют, например:

НайденныеОшибки.Колонки.Добавить("Номер"); // для совместимости

7.3. Большие комментарии или комментарии к фрагменту кода пишутся перед комментируемым кодом в отдельной строке. Текст выравнивается по левой границе комментируемого фрагмента. Между символами комментария "//" и текстом комментария должен быть пробел.

// Инициализируем переменные для выполнения расчетов,

// которые выполняются далее по тексту модуля.

ТекущаяДата = ОбщегоНазначения.ПолучитьРабочуюДату();

ТекущийГод = Год(ТекущаяДата);

ТекущийМесяц = Месяц(ТекущаяДата);

ТекущаяНеделя = НеделяГода(ТекущаяДата);

ТекущийДень = День(ТекущаяДата);

8. Тексты больших процедур и функций можно разбивать на отдельные сворачиваемые области. При этом имена областей должны удовлетворять требованиям стандарта [Правила образования имен переменных](https://its.1c.ua/db/content/v8std/src/300/100/i8100454.htm?_=1580137113)

Для автоматического форматирования кода можно воспользоваться обработкой [автоформатирования кода и локализации](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%90%D0%B2%D1%82%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%9A%D0%BE%D0%B4%D0%B0%D0%98%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F/%D0%90%D0%B2%D1%82%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%9A%D0%BE%D0%B4%D0%B0%D0%98%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F.zip).

См. также

* [Правила создания общих модулей](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1580137113)
* [Структура модуля](https://its.1c.ua/db/content/v8std/src/300/100/i8100455.htm?_=1580137113)

### Структура модуля

#std455

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. В программном модуле (общие модули, модули объектов, модули менеджеров объектов, модули форм, команд и т.п.) в общем случае могут присутствовать следующие разделы в приведенной ниже последовательности:

* заголовок модуля
* раздел описания переменных
* экспортные процедуры и функции модуля, составляющие его программный интерфейс
* обработчики событий объекта (формы)
* служебные процедуры и функции модуля
* раздел инициализации

Некоторые разделы могут присутствовать только в модулях определенного вида. Например, обработчики событий элементов форм могут присутствовать только в модулях форм, а раздел описания переменных и раздел инициализации не могут быть определены в неглобальных общих модулях, модулях менеджеров объектов, наборов записей, значений констант и модуле сеанса.

Требование о разделении кода модуля на разделы призвано повысить читаемость кода и упростить внесение изменений в код разными авторами (разработчиками) как при коллективной разработке, так и при доработке прикладных решений на конкретных внедрениях.

1.2. Объемные разделы модулей рекомендуется разбивать на подразделы по функциональному признаку.

1.3. Разделы и подразделы оформляются в виде областей. При этом имена областей должны удовлетворять требованиям стандарта [Правила образования имен переменных](https://its.1c.ua/db/content/v8std/src/300/100/i8100454.htm?_=1580137113)

1.4. Шаблон (заготовка для копирования) разделов для общих модулей:

**русск.**

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

**англ.**

#Region Public

// Enter code here.

#EndRegion

#Region Internal

// Enter code here.

#EndRegion

#Region Private

// Enter code here.

#EndRegion

* Раздел **«Программный интерфейс»** содержит экспортные процедуры и функции, предназначенные для использования другими объектами конфигурации или другими программами (например, через внешнее соединение).
* Раздел «**Служебный программный интерфейс**»  предназначен для модулей, которые являются частью некоторой функциональной подсистемы. В нем должны быть размещены экспортные процедуры и функции, которые допустимо вызывать только из других функциональных подсистем этой же библиотеки.
* Раздел **«Служебные процедуры и функции»** содержит процедуры и функции, составляющие внутреннюю реализацию общего модуля. В тех случаях, когда общий модуль является частью некоторой [функциональной подсистемы](https://its.1c.ua/db/content/v8std/src/100/100/i8100543.htm?_=1580137113), включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

Для объемных общих модулей рекомендуется разбивать этот раздел на подразделы, по функциональному признаку. Например:

**русск.**

#Область ОбновлениеИнформационнойБазы

// Код процедур и функций

#КонецОбласти

**англ.**

#Region InfobaseUpdate

// Enter code here.

#EndRegion

1.5. Шаблон оформления разделов для модулей объектов, менеджеров, наборов записей, обработок, отчетов и т.п.:

**русск.**

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

**англ.**

#Region Public

// Enter code here.

#EndRegion

#Region EventHandlers

// Enter code here.

#EndRegion

#Region Internal

// Enter code here.

#EndRegion

#Region Private

// Enter code here.

#EndRegion

* Раздел **«Программный интерфейс»** содержит экспортные процедуры и функции, предназначенные для использования в других модулях конфигурации или другими программами (например, через внешнее соединение). Не следует в этот раздел помещать экспортные функции и процедуры, которые предназначены для вызова исключительно из модулей самого объекта, его форм и команд. Например, процедуры заполнения табличной части документа, которые вызываются из обработки заполнения в модуле объекта и из формы документа в обработчике команды формы не являются программным интерфейсом модуля объекта, т.к. вызываются только в самом модуле и из форм этого же объекта. Их следует размещать в разделе «Служебные процедуры и функции».
* Раздел **«Обработчики событий»** содержит обработчики событий модуля объекта (**ПриЗаписи**, **ПриПроведении** и др.)
* Раздел «**Служебный программный интерфейс**» имеет такое же предназначение, как и в общих модулях.
* Раздел **«Служебные процедуры и функции»** имеет такое же предназначение, как и в общих модулях.

1.6. Шаблон оформления разделов для модулей форм:

**русск.**

#Область ОбработчикиСобытийФормы

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытийЭлементовШапкиФормы

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытийЭлементовТаблицыФормы<ИмяТаблицыФормы>

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиКомандФормы

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

**англ.**

#Region FormEventHandlers

// Enter code here.

#EndRegion

#Region FormHeaderItemsEventHandlers

// Enter code here.

#EndRegion

#Region FormTableItemsEventHandlers

// Enter code here.

#EndRegion

#Region FormCommandsEventHandlers

// Enter code here.

#EndRegion

#Region Private

// Enter code here.

#EndRegion

* Раздел **«Обработчики событий формы»** содержит процедуры-обработчики событий формы: **ПриСозданииНаСервере**, **ПриОткрытии** и т.п.
* Раздел **«Обработчики событий элементов шапки формы»** содержит процедуры-обработчики элементов, расположенных в основной части формы (все, что не связано с таблицами на форме).
* В разделах **«Обработчики событий элементов таблицы формы <имя таблицы формы>»** размещаются процедуры-обработчики таблиц формы и элементов таблиц. Для процедур-обработчиков каждой таблицы должен быть создан свой раздел.
* Раздел **«Обработчики команд формы»** содержит процедуры-обработчики команд формы (имена которых задаются в свойстве Действие команд формы).
* Раздел **«Служебные процедуры и функции»** имеет такое же предназначение, что и в общих модулях.

См. также: [*Правила создания модулей форм*](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100630.htm?_=1580137113)

1.7. Шаблон оформления разделов для модулей команд:

**русск.**

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

**англ.**

#Region EventHandlers

// Enter code here.

#EndRegion

#Region Private

// Enter code here.

#EndRegion

* Раздел **«Обработчики событий»** содержит процедуру-обработчик команды **ОбработкаКоманды**.
* Раздел **«Служебные процедуры и функции»** имеет такое же предназначение, что и в общих модулях.

1.8. В модуле не должно быть пустых областей.

2. Общие требования к разделам программных модулей.

2.1. **Заголовок модуля** представляет собой комментарий в самом начале модуля. В заголовке модуля приводится его краткое описание и условия применения.  
Например:

////////////////////////////////////////////////////////////////////////////////

// Клиентские процедуры и функции общего назначения:

// - для работы со списками в формах;

// - для работы с журналом регистрации;

// - для обработки действий пользователя в процессе редактирования

// многострочного текста, например комментария в документах;

// - прочее.

//

////////////////////////////////////////////////////////////////////////////////

Для модулей форм в заголовке рекомендуется размещать описание параметров формы.

2.2. **Раздел описания переменных**. Имена переменных назначаются согласно [общим правилам образования имен переменных](https://its.1c.ua/db/content/v8std/src/300/100/i8100454.htm?_=1580137113), а их использование описывается в статье [Использование глобальных переменных в программных модулях](https://its.1c.ua/db/content/v8std/src/300/200/i8100639.htm?_=1580137113).

Все переменные модуля должны быть снабжены комментарием, достаточным для понимания их назначения. Комментарий рекомендуется размещать в той же строке, где объявляется переменная.  
Пример:

**русск.**

#Область ОписаниеПеременных

Перем ВалютаУчета;

Перем АдресПоддержки;

...

#КонецОбласти

**англ.**

#Region Variables

Var PresentationCurrency;

Var SupportEmail;

...

#EndRegion

2.3. **Программный интерфейс**. Экспортные процедуры и функции, составляющие его программный интерфейс, размещаются сразу же после описания переменных. Такие процедуры и функции предназначены для использования другими объектами конфигурации или другими программами (например, через внешнее соединение), поэтому должны быть расположены в модуле на "видном месте".

См. также: [*Описание процедур и функций*](https://its.1c.ua/db/content/v8std/src/300/100/i8100453.htm?_=1580137113)

2.4.1. **Обработчики событий формы, команд и элементов формы**. Перед служебными процедурами и функциями в модуле формы располагаются обработчики событий формы, а также обработчики событий команд и элементов формы.

|  |
| --- |
| Методическая рекомендация (полезный совет)  2.4.2. Рекомендуется обработчики одного элемента формы располагать вместе, придерживаясь, при этом, порядка их следования в панели свойств редактора формы в конфигураторе.  2.4.3. У каждого события должна быть назначена своя процедура-обработчик. Если одинаковые действия должны выполняться при возникновении событий в разных элементах формы следует:   * создать отдельную процедуру (функцию), выполняющую необходимые действия * для каждого элемента формы создать отдельный обработчик с именем, назначаемым по умолчанию * из каждого обработчика вызвать требуемую процедуру (функцию).   Например, неправильно:  &НаКлиенте  Процедура ПоИсполнителюПриИзменении(Элемент)  ПараметрыОтбора = Новый Соответствие();  ПараметрыОтбора.Вставить("ПоАвтору", ПоАвтору);  ПараметрыОтбора.Вставить("ПоИсполнителю", ПоИсполнителю);  УстановитьОтборСписка(Список, ПараметрыОтбора);  КонецПроцедуры  &НаКлиенте  Процедура ПоАвторуПриИзменении(Элемент)  ПоИсполнителюПриИзменении(Неопределено);  КонецПроцедуры  правильно:  &НаКлиенте  Процедура ПоИсполнителюПриИзменении(Элемент)  УстановитьОтбор();  КонецПроцедуры  &НаКлиенте  Процедура ПоАвторуПриИзменении(Элемент)  УстановитьОтбор();  КонецПроцедуры  &НаСервере  Процедура УстановитьОтбор()  ПараметрыОтбора = Новый Соответствие();  ПараметрыОтбора.Вставить("ПоАвтору", ПоАвтору);  ПараметрыОтбора.Вставить("ПоИсполнителю", ПоИсполнителю);  УстановитьОтборСписка(Список, ПараметрыОтбора);  КонецПроцедуры  Это требование обусловлено тем, что логически процедуры-обработчики событий не предназначены для использования в коде модуля, а вызываются непосредственно платформой. Смешение же этих двух сценариев в одной процедуре неоправданно усложняет ее логику и снижает ее устойчивость (вместо одного предусмотренного сценария вызова - по событию из платформы - код процедуры должен рассчитывать и на другие "прямые" вызовы из кода). |

2.5. **Обработчики событий модулей объекта и менеджера объекта**размещаются после раздела с программным интерфейсом, но до служебных процедур и функций модуля.

|  |
| --- |
| Методическая рекомендация (полезный совет)  2.5.1. Рекомендуется располагать обработчики, придерживаясь порядка их следования в описании встроенного языка. |

2.6. **Служебные процедуры и функции модуля**, которые не являются обработчиками событий, а составляют внутреннюю реализацию модуля, размещаются в модуле следом за обработчиками событий.

В тех случаях когда общий модуль является частью некоторой функциональной подсистемы, включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

Процедуры и функции, связанные между собой по характеру или по логике работы рекомендуется располагать вместе. В модулях форм не рекомендуется явно группировать процедуры и функции модуля на серверные, клиентские и функции без контекста, так как такое «технологическое» упорядочивание затрудняет понимание логики модуля, отвлекая внимание разработчика на детали ее реализации.

2.7. **Раздел инициализации** содержит операторы, инициализирующие переменные модуля или объект (форму).  
Например:

**русск.**

#Область Инициализация

АдресПоддержки = "v8@1c.ru";

ВыполнитьИнициализацию();

...

#КонецОбласти

**англ.**

#Region Initialize

SupportEmail = "v8@1c.ru";

Ctor();

...

#EndRegion

Для оформления разделов кода в виде областей рекомендуется воспользоваться [приложенной обработкой](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D1%80%D1%82%D0%B0%D1%86%D0%B8%D1%8F%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B5%D0%B9%D0%94%D0%BB%D1%8F%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D0%9E%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D0%B9/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D1%80%D1%82%D0%B0%D1%86%D0%B8%D1%8F%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B5%D0%B9%D0%94%D0%BB%D1%8F%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D0%9E%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D0%B9.zip).

### Имена процедур и функций

#std647

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Правильный выбор имен процедур и функций очень важен для повышения читаемости кода. В большинстве случаев хорошо выбранное имя процедуры в сочетании с правильно подобранными именами параметров избавляют от необходимости ее как-то дополнительно описывать. В ряде случаев, сложности в выборе имени процедуры и (или) ее параметров свидетельствуют о неправильной архитектуре программного кода. И наоборот, если "самодокументирующееся" имя придумать легко, значит процедура спроектирована правильно.

См. также: [*Описание процедур и функций*](https://its.1c.ua/db/content/v8std/src/300/100/i8100453.htm?_=1580137113)

2. Имена процедур, функций и [формальных параметров](https://its.1c.ua/db/content/v8std/src/300/100/i8100640.htm?_=1580137113) следует образовывать от терминов предметной области таким образом, чтобы из имени было понятно назначение. Следует стремиться к тому, чтобы имена были "говорящими" (документировали сами себя).  
Например, неправильно:

Функция ВыполнитьПроверку(Параметр1, Рекв, ТЗ)

Функция ПолучитьМассивыРеквизитов(ХозяйственнаяОперация, МассивВсехРеквизитов, МассивРеквизитовОперации)

Правильно:

Функция РеквизитОбъектаЗаданногоТипа(Объект, ИмяРеквизита, ТипЗначения)

Функция ЗаполнитьИменаРеквизитовПоХозяйственнойОперации(ХозяйственнаяОперация, ИменаВсеРеквизиты, ИменаРеквизитыОперации)

3. Имена следует образовывать путем удаления пробелов между словами. При этом, каждое слово в имени пишется с прописной буквы. Предлоги и местоимения из одной буквы также пишутся прописными буквами.

|  |
| --- |
| Методическая рекомендация (полезный совет)  4. Не рекомендуется в названиях процедур и функций описывать типы принимаемых параметров и (или) возвращаемых значений.  Например, неправильно:  Функция ПолучитьМассивРолейСПравомДобавления()  Функция ПолучитьСтруктуруДополнительныхНастроек()  Правильно:  Функция ИменаРолейСПравомДобавления()  Функция ДополнительныеНастройки()  Эта рекомендация справедлива в большинстве случаев за редким исключением, когда без описания типа возвращаемого значения не ясно назначение самой процедуры или функции.  5. Имена процедур в общем случае, следует образовывать от неопределенной формы глагола, от сути выполняемого действия, например:  Неправильно:  Процедура ЗагрузкаКонтрагента()  Правильно:  Процедура ЗагрузитьКонтрагента()  6.1. Имена функций в общем случае следует образовывать от описания возвращаемого значения.  Неправильно:  Функция ПолучитьПолноеИмяПользователя()  Функция СоздатьПараметрыЗаполненияЦенПоставщика()  Функция ОпределитьДатуНачалаСеанса()  Правильно:  Функция ПолноеИмяПользователя()  Функция НовыеПараметрыЗаполненияЦенПоставщика()  Функция ДатаНачалаСеанса()  6.2. Если функция предназначена для создания какого-либо объекта, то рекомендуется в ее имени использовать слово "Новый". Например, Неправильно:  Функция ДобавитьПолеФормы()  Функция СоздатьЭлементСправочникаФайлы()  Функция ПолучитьТаблицуКоманд()  Правильно:  Функция НовоеПолеФормы()  Функция НовыйЭлементСправочникаФайлы()  Функция НоваяТаблицаКоманд()  6.3. Если функция выполняет проверку какого-то условия, то ее имя рекомендуется начитать со слова "Это" или использовать причастия. Неправильно:  Функция ПроверитьПроведенностьДокумента()  Функция ПроверитьИзменениеРеквизитовДокумента()  Функция ВнешняяЗадача()  Правильно:  Функция ДокументПроведен()  Функция РеквизитыДокументыИзменены()  Функция ЭтоВнешняяЗадача()  6.4. В имени функции рекомендуется использовать глаголы в неопределенной форме в тех случаях, когда для понимания назначения функции важно, каким образом было получено возвращаемое значение. Например:  Функция ВыбратьДанныеПоПравилу(Правило, ПользовательскиеНастройки)  Функция ПреобразоватьДанныеПоПравилу(НаборыДанных, ПараметрыПреобразования)  6.5. Если выполнение функции предполагает, прежде всего, какое-либо действие, и при этом возврат значения не является ее основной задачей (например, это признак успешно выполненного действия), то имена таких функций следует образовывать от неопределенной формы глагола, как и для процедур. Например:  Функция РазрешитьРедактированиеРеквизитовОбъекта(Форма) |

### Описание процедур и функций

#std453

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Описание [процедур и функций](https://its.1c.ua/db/content/v8std/src/300/100/i8100647.htm?_=1580137113) рекомендуется выполнять в виде комментария к ним. Необходимость комментирования отдельных участков кода процедур и функций должна определяться разработчиком исходя из сложности и нестандартности конкретного участка кода.

При разработке на платформе 1С:Предприятие 8.3 текст комментария также выводится в контекстной подсказке процедур, функций и их параметров. Подробнее см. раздел «[Контекстная подсказка при вводе текстов модулей](http://its.1c.ru/db/v83doc#bookmark:dev:TI000000942)» главы 27 «Инструменты разработки» в документации к платформе.

При разработке в **1C:Enterprise Development Tools** текст комментария также используется для уточнения типизации параметров и возвращаемого значения процедур и функций, и тем самым помогает выявлять ошибки кодирования на этапе разработки.

2. Обязательного комментирования требуют процедуры и функции входящие в [программный интерфейс модулей](https://its.1c.ua/db/content/v8std/src/300/100/i8100455.htm?_=1580137113) - такие процедуры и функции предназначены для использования в других функциональных подсистемах (или в других приложениях), за которые могут отвечать другие разработчики, поэтому они должны быть хорошо документированы.

См. также: [Ограничения на использование экспортных процедур и функций](https://its.1c.ua/db/content/v8std/src/300/300/i8100544.htm?_=1580137113) и [*Использование экспортных процедур и функций в модулях форм*](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100630.htm?_=1580137113)

3. Прочие процедуры и функции (в том числе обработчики событий модулей форм, объектов, наборов записей, менеджеров значений и т.п.) рекомендуется комментировать, если требуется пояснить назначение процедуры (функции) или особенности её работы. Также рекомендуется описывать причины невыполнения некоторых действий, если они кажутся неочевидными для данной процедуры или функции.

Но если процедура (функция) не сложна для понимания и ее назначение и порядок работы следуют из ее названия и имен формальных параметров, комментарий допускается не писать.

4.  Следует избегать комментариев, не дающих дополнительных пояснений о работе не-экспортной процедуры (функции).  
Например, неправильно:

// Процедура - обработчик события "ПриОткрытии" формы

//

&НаКлиенте

Процедура ПриОткрытии()

// Процедура-обработчик команды "Рассчитать"

//

&НаКлиенте

Процедура Рассчитать()

// Процедура-обработчик события "ПриИзменении" элемента формы "РедактированиеТолькоВДиалоге"

//

&НаКлиенте

Процедура РедактированиеТолькоВДиалогеПриИзменении(Элемент)

В этих примерах комментарии избыточны, так как из названий процедур очевидно, что это обработчики событий. А с их описанием и назначением параметров можно ознакомиться в синтакс-помощнике.

// Функция возвращает статью движения денежных средств по данным документа

Функция СтатьяДвиженияДенежныхСредств(ДанныеДокумента)

Этот комментарий не дает никакой дополнительной информации о функции.

5. Комментарий размещается перед объявлением процедуры (функции) и имеет следующий вид.  
  
5.1. **Секция "Описание"**(англ. **"Description"**)  содержит описание назначения процедуры (функции), достаточное для понимания сценариев ее использования без просмотра ее исходного кода. Также может содержать краткое описание принципов работы и перекрестные ссылки на связанные процедуры и функции.

Может быть единственной секцией для процедур без параметров. Описание не должно совпадать с именем процедуры (функции). Для процедур и функций секция должна начинаться с глагола. Для функций это, как правило: «Возвращает…». В тех случаях, когда возвращаемый результат является не основным в работе функции, – то с основного действия, например: «Проверяет…», «Сравнивает…», «Вычисляет…» и т.п. Не рекомендуется начинать описание с избыточных слов «Процедура...», «Функция...», а также с имени самой процедуры (функции), от удаления которых смысл не меняется.

Например, неправильно:

// Конструктор объекта WSПрокси.

// ...

Функция WSПрокси(ПараметрыПрокси) Экспорт

// Функция СтрокаТаблицыЗначенийВСтруктуру создает структуру со свойствами, соответствующими...

Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт

Правильно:

// Создает прокси на основе определения веб-сервиса и связывает

// его с точкой подключения веб-сервиса.

// В дополнении к платформенному конструктору Новый WSПрокси:

// - включает в себя вызов конструктора WSОпределения;

// - на время сеанса кэширует файл WSDL для оптимизации частых обращений к веб-сервису;

// - не требует явного указания ИнтернетПрокси (он подставляется автоматически, если настроен);

// - выполняет быструю проверку доступности веб-сервиса с помощью операции Ping.

// ...

Функция WSПрокси(ПараметрыПрокси) Экспорт

// Создает структуру со свойствами, соответствующими...

Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт

5.2. **Секция "Параметры"**(англ.**"Parameters"**)описывает [параметры процедуры (функции)](https://its.1c.ua/db/content/v8std/src/300/100/i8100640.htm?_=1580137113). Если их нет, секция пропускается. Предваряется строкой "Параметры:", затем с новой строки размещаются описания всех параметров.

5.2.1. **Описание параметра** начинается с новой строки, далее имя параметра, затем дефис и список типов (\*), далее дефис и текстовое описание параметра.

**Имя параметра** необходимо стремиться выбирать таким образом, чтобы его назначение было понятно в контексте функции без дополнительных пояснений

**Описание типа** является обязательным. Тип может быть описан явно, при этом может быть указан или один тип или список типов. Под «списком типов» подразумеваются имена типов, разделенные запятыми. Имя типа может быть простым (в одно слово) или составным - в два слова, разделенных точкой.  
Например: **Строка, Структура, Произвольный, СправочникСсылка.Сотрудники**.

Кроме того, не следует использовать в качестве типов значений несуществующие в платформе типы.

Например, неправильно:

// КоллекцияСтрок - КоллекцияЗначений – коллекция для сравнения;

Правильно:

// КоллекцияСтрок – ТаблицаЗначений, Массив, СписокЗначений – Элемент для сравнения.

**Текстовое описание** параметра рекомендуется заполнять в том случае, когда только имени параметра в контексте функции не достаточно для понимания его назначения, либо требуется дать дополнительную информацию о типе, поясняющие назначение параметра, а также может приводиться наглядный пример с ожидаемым значением параметра.

Например, неправильно:

// Проверяет, что переданные адреса включены в задачу. Если проверка не проходит – генерируется исключение.  
//  
// Параметры:  
//  Строка - Строка - строка, содержащая электронные адреса  
//  ЗадачаСсылка  - ЗадачаСсылка.ЗадачаИсполнителя – проверяемая задача  
//  
Процедура ПроверитьАдресаЗадачи(Адреса, ЗадачаИсполнителя)

Правильно:

// Проверяет, что переданные адреса включены в задачу. Если проверка не проходит – генерируется исключение.  
//  
// Параметры:  
//  Адреса - Строка - содержит электронные адреса, разделенные запятой. Например, "support@mycorp.ru,v8@localdomain"  
//  Задача - ЗадачаСсылка.ЗадачаИсполнителя  
//  
Процедура ПроверитьАдресаЗадачи(Адреса, ЗадачаИсполнителя)

В данном примере текстовое описание для параметра «Адреса» нужно чтобы

* указать правило передачи нескольких адресов (через зяпятую)
* привести пример

Текстовое описание для параметра «Задача» не нужно.

5.2.2. Для параметров типа **Структура** и **ТаблицаЗначений** также задается описание их свойств и колонок, которые начинаются с новой строки и предваряются символом \*.  
Например:

// Параметры:

// СтатусыСерий - ТаблицаЗначений - таблица с колонками:

// \* Серия - СправочникСсылка.СерииНоменклатуры - если серия указана и она может

// использоваться с новым значением номенклатуры на указанном складе,

// то возвращается переданное значение; если нет - пустая ссылка

// \* СтатусУказанияСерий - Число - если серии указываются в ТЧ "Товары", то

// возвращается рассчитанный статус, если для переданной

// номенклатуры/склада серии не используется - возвращается 0

// иначе возвращается переданный статус.

5.2.3. Для параметров типа **Массив** следует указывать тип элементов с помощью ключевого слова **"из"** (англ. **"contains"**):

Например, неправильно:

// МассивПеренаправленныхЗадач - Массив - массив перенаправленных задач.

// МассивПеренаправленныхЗадач - Массив - задачи (ЗадачаСсылка.ЗадачаИсполнителя), перенаправленные другому исполнителю.

Правильно:

// ПеренаправленныеЗадачи - Массив из ЗадачаСсылка.ЗадачаИсполнителя  
   
// СведенияОбОбновлении - Массив из Структура - содержит:

// \* КодАдресногоОбъекта - Строка

// \* Наименование - Строка

// \* Индекс - Строка

// \* ОбновлениеДоступно - Булево

//

В описании массивов, структур и таблиц значений могут быть вложенные описания, при этом перед именами вложенных свойств число звездочек увеличивается: для первого уровня вложенности 2 звездочки, для второго 3 и т.д.

// Параметры:

// СведенияОбОбновлении - Массив из Структура - содержит:

// \* КодАдресногоОбъекта - Структура - содержит:

// \*\* КодРегиона - Число - код региона (длина - 2).

// \*\* КодНаселенногоПункта - Число - код населенного пункта (длина - 3).

// \*\* КодУлицы - Число - код улицы (длина - 4).

// \* Наименование - Строка

// \* ОбновлениеДоступно - Булево

//

5.2.4. Также для каждого параметра можно задать одно или несколько дополнительных описаний типов параметра. Каждое дополнительное описание начинается с новой строки, затем обязательный дефис, далее список типов параметра далее дефис и текстовое описание.  
Например:

// Параметры:

// Реквизиты - Строка - имена реквизитов, перечисленные через запятую.

// Например, "Код, Наименование, Родитель".

// - Структура, ФиксированнаяСтруктура - в качестве ключа передается

// псевдоним поля для возвращаемой структуры с результатом,

// а в качестве значения (опционально) фактическое имя поля в таблице.

// Если значение не определено, то имя поля берется из ключа.

// - Массив, ФиксированныйМассив - имена реквизитов (Строка).

5.2.5. Описание также могут быть заданы с помощью ссылки на функцию-конструктор в формате **"см. ПутьКФункции"** (англ **"see MethodPath"**).

Например:

// ПараметрыУказанияСерий - см. НоменклатураКлиентСервер.ПараметрыУказанияСерий

// РеквизитыКомпонент - Массив из см. ВнешниеКомпонентыСлужебный.РеквизитыКомпоненты

5.3. **Секция "Возвращаемое значение"**(англ. **"Returns"**) описывает тип и содержание возвращаемого значения функции. Для процедур эта секция отсутствует. Предваряется строкой "Возвращаемое значение:". Затем с новой строки тип возвращаемого значения, дефис и текст описания. При использовании возвращаемого значения составного типа следует каждый тип писать с новой строки и с дефиса. Например:

// Возвращаемое значение:

// Строка

// Возвращаемое значение:

// Булево - Истина, если хотя бы одна из переданных ролей доступна текущему пользователю, либо у него есть административные права.

// Возвращаемое значение:

// - ЛюбаяСсылка - ссылка на предопределенный элемент.

// - Неопределено - если предопределенный элемент есть в метаданных, но не создан в ИБ.

// Возвращаемое значение:

// - СправочникСсылка.Пользователи

// - СправочникСсылка.ВнешниеПользователи

Текстовое описание возвращаемого значения рекомендуется заполнять в том случае, когда только одного описания функции не достаточно, либо требуется дать дополнительную информацию о типе, например, о составе свойств или колонок возвращаемого значения. Также может быть приведен пример с ожидаемым значением возвращаемого значения, либо сквозной пример размещается в секции "Пример" ниже.

Для возвращаемых значений также действуют требования п.5.2.2 и 5.2.3.

5.4. **Секция "Пример"**(англ. **"Example"**) содержит пример использования процедуры, или функции. Предваряется строкой "Пример:". Далее с новой строки пример использования. Имя процедуры (функции) следует писать вместе с именем общего модуля, в котором она расположена. Из примера должно быть понятно, что передается на входе и что возвращается на выходе.  
Например, неправильно:

// Пример:

// ПодставитьПараметрыВСтроку(ШаблонСтроки, СтрокаЗамены);

Правильно:

// Пример:

// СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru='%1 пошел в %2'"), "Вася", "Зоопарк") = "Вася пошел в Зоопарк".

5.4.1. В [переопределяемых модулях](https://its.1c.ua/db/content/v8std/src/800/i8100554.htm?_=1580137113) в **секции "Пример"** следует размещать пример реализации переопределяемой процедуры, а не пример ее вызова. Например, для процедуры ПриОпределенииОбщихПараметровБазовойФункциональности(ОбщиеПараметры):

// Пример:

// ОбщиеПараметры.МинимальноНеобходимаяВерсияПлатформы = "8.3.4.365";

// ОбщиеПараметры.РекомендуемыйОбъемОперативнойПамяти = 2;

5.5. В редких случаях, когда сразу несколько параметров имеют дополнительные типы, рекомендуется добавить **секцию "Варианты вызова"** (англ. **"Сall options"**), в которой дать описания наиболее частых или всех возможных вариантов вызова функции с различными комбинациями типов параметров. Секция начинается фразой "Варианты вызова:" с новой строки, затем идут описания вариантов, каждое начинается с новой строки. Каждый вариант вызова представляется в виде имени функции со списком типов, перечисленных через запятую в круглых скобках, затем следует дефис и текстовое описание варианта.

Например:

// ...

//

// Параметры:

// Параметр1 - Тип11, Тип12 - ...

// Параметр2 - Тип21, Тип22, Тип23 - ...

//

// Варианты вызова:

// УниверсальнаяПроцедура(Тип11, Тип21) - описание ...

// УниверсальнаяПроцедура(Тип12, Тип22) - описание ...

// УниверсальнаяПроцедура(Тип11, Тип23) - описание ...

//

Процедура УниверсальнаяПроцедура(Параметр1, Параметр2) Экспорт

5.6. В любом месте документирующего комментария можно добавить переход к другим объектам конфигурации, процедурам и функциям (в частности, для перехода к [функциям-конструкторам структур](https://its.1c.ua/db/content/v8std/src/300/100/i8100641.htm?_=1580137113)). При использовании **1C:Enterprise Development Tools**среда оформит такие переходы в виде гиперссылки.  
Например:

// Описание универсальной процедуры.

//

// См. УправлениеДоступом.ЗаполнитьНаборыЗначенийДоступа

//

// Параметры:

// Параметр1 – Произвольный – описание параметра см. Справочник.Контрагенты.

//

Процедура УниверсальнаяПроцедура(Параметр1)

5.7. В случаях когда возникает необходимость отметить процедуру (функцию) как устаревшую, в первой строке ее описания размещается слово **"Устарела"** (англ. **"Deprecated"**)..  
Например:

// Устарела. Следует использовать новую см. ОбщегоНазначения.ЕстьРоль

// ...

Функция РолиДоступны(ИменаРолей) Экспорт

6. Если требуется прокомментировать процедуру или функцию с директивой компиляции, то вначале следует размещать комментарий, а затем -  
директиву компиляции. Например:

// Процедура - обработчик события "ПриСозданииНаСервере" формы.

// Обрабатывает параметры формы и заполняет реквизиты формы значениями.

// А также выполняет следующие действия:

// ...

//

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

Такой стиль размещения комментария позволяет в первую очередь обращать внимание на определение функции и директиву компиляции, а потом - на комментарий, который может занимать достаточно большое количество строк.

7. Код процедур и функций должен отделяться друг от друга в тексте модуля пустыми строками.

#### Примеры описания процедур и функций

Пример описания функции с одним параметром:

// Определяет доступность ролей ИменаРолей текущему пользователю,

// а также доступность административных прав.

//

// Параметры:

// ИменаРолей - Строка - имена ролей, доступность которых проверяется, разделенные запятыми.

//

// Возвращаемое значение:

// Булево - Истина, если хотя бы одна из переданных ролей доступна текущему пользователю,

// либо у него есть административные права.

//

// Пример:

// Если РолиДоступны("ИспользованиеРассылокОтчетов,ОтправкаПоПочте") Тогда ...

//

Функция РолиДоступны(ИменаРолей) Экспорт

Пример описания процедуры без параметров:

// В обработчике события ПередЗаписью документа выполняется;

// - очистка табличной части услуги, в случае если указан договор с комиссионером;

// - проверка заполнения реквизита ЕдиницаИзмеренияМест табл. части Товары;

// - синхронизация с "подчиненным" счетом-фактурой;

// - заполнение склада и заказа покупателя в табличных частях Товары и ВозвратнаяТара;

// - удаление неиспользуемых строк табличной части "Серийные номера";

// - заполнение переменной модуля объекта УдалятьДвижение.

//

Процедура ПередЗаписью()

КонецПроцедуры

|  |
| --- |
| Для автоматического упорядочивания комментариев к процедурам или функциям с директивами компиляции можно воспользоваться приложенной обработкой [**ФорматированиеДирективКомпиляции.epf**](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%94%D0%B8%D1%80%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D0%B8/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%94%D0%B8%D1%80%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D0%B8.zip).  Для этого необходимо:   1. Выгрузить модули конфигурации (команда меню Конфигурация -> Выгрузить файлы конфигурации...) 2. Открыть обработку в режиме **1С:Предприятие** и указать каталог, в который были выгружены модули - далее нажать кнопку "Форматировать" 3. Загрузить модули в конфигурацию (команда меню Конфигурация -> Загрузить файлы конфигурации...) |

### Параметры процедур и функций

#std640

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При объявлении формальных параметров процедур и функций (далее по тексту: функций) необходимо придерживаться [общих правил образования имен переменных](https://its.1c.ua/db/content/v8std/src/300/100/i8100454.htm?_=1580137113). В частности, имена параметров следует образовывать от терминов предметной области таким образом, чтобы из имени параметра было понятно его назначение.

2. Не следует использовать вместо параметров функций другие средства конфигурирования ([переменные модулей](https://its.1c.ua/db/content/v8std/src/300/200/i8100639.htm?_=1580137113), реквизиты формы и т.п.)

3. Параметры в функции должны идти в логической последовательности. Рекомендуется располагать параметры по принципу от общего к частному.  
Например, неправильно:

Процедура ПересчитатьСуммуДокумента(ИмяПоляСумма, ДокументОбъект, СуммаВключаетНДС = Истина)

Процедура ПоменятьЦветПоляФормы(Цвет, ИмяПоля, Форма)

правильно сначала расположить основные параметры **ДокументОбъект** и**Форма**:

Процедура ПересчитатьСуммуДокумента(ДокументОбъект, ИмяПоляСумма, СуммаВключаетНДС = Истина)

Процедура ПоменятьЦветПоляФормы(Форма, ИмяПоля, Цвет)

4. Необязательные параметры (параметры со значениями по умолчанию) должны располагаться после обязательных параметров (без значений по умолчанию).  
Например:

Функция КурсВалютыНаДату(Валюта, Дата = Неопределено) Экспорт

5. Не рекомендуется объявлять в функциях много параметров (нужно ориентироваться на количество не более семи параметров), при этом не должно быть много параметров со значениями по умолчанию (нужно ориентироваться на количество не более трех таких параметров). В противном случае, читаемость вызывающего кода сильно снижается. Например, можно легко ошибиться в количестве запятых при передаче необязательных параметров.

При необходимости передавать в функцию большое число параметров рекомендуется:

* группировать однотипные параметры в один или несколько составных параметров типа **Структура**. Например, в структуры могут быть объединены параметры, описывающие состав и значения полей некоторого объекта (**ДанныеЗаполнения**, **ПараметрыПроведения, ДанныеФайла** и т.п.);
* либо полностью пересмотреть логику работы функции, например, разделив ее на несколько разных, более простых функций.

Например, неправильно:

// Добавляет новое поле на форму, инициализирует его значениями по умолчанию.

Функция ДобавитьПолеФормы(ИмяПоля,

Заголовок = Неопределено,

ОбработчикПриИзменении = "",

ОбработчикНачалоВыбора = "",

ШиринаПоля,

ЦветФона = Неопределено,

ЦветФонаЗаголовка = Неопределено,

Родитель = Неопределено,

КартинкаШапки = Неопределено,

ПутьКДанным = Неопределено,

ТолькоПросмотрПоля = Ложь,

СвязиПараметровВыбора = Неопределено)

…

КонецФункции

// вызывающий код

НовоеПоле = ДобавитьПолеФормы("СтараяЦена", НСтр("ru='Цена'"),,, 12, ЦветФона, ЦветЗаголовка, НоваяГруппа,,,Истина);

НовоеПоле.ЦветТекста = WebЦвета.Серый;

Правильно пересмотреть логику работы функций, оставив в ней только один ключевой параметр **ИмяПоля**:

// Добавляет новое поле на форму, инициализирует его значениями по умолчанию.

Функция НовоеПолеФормы(ИмяПоля)

…

КонецФункции

// вызывающий код

НовоеПоле = НовоеПолеФормы("СтараяЦена");

НовоеПоле.Заголовок = НСтр("ru='Цена'");

НовоеПоле.ЦветФона = ЦветФона;

НовоеПоле.ЦветТекста = WebЦвета.Серый;

НовоеПоле…. = …

…

Другой пример. Неправильно:

// Создает элемент справочника "Номенклатура"

Процедура СоздатьЭлементНоменклатуры(Наименование, ТоварУслуга, ЕдиницаИзмерения, ВесНетто, ПроверятьУникальность = Истина)

…

КонецПроцедуры

Правильно сгруппировать параметры, описывающие значения реквизитов номенклатуры, в структуру **ЗначенияРеквизитов**:

// Создает элемент справочника "Номенклатура"

Процедура СоздатьЭлементНоменклатуры(ЗначенияРеквизитов, ПроверятьУникальность = Истина)

…

КонецПроцедуры

6. При вызове функций необходимо избегать громоздких конструкций, которые приводят к снижению читаемости кода, увеличивают вероятность ошибок и затрудняют отладку.  
В частности:

6.1. Не рекомендуется при передаче параметров в одну функцию применять вложенные вызовы других функций.  
Неправильно:

СтруктураВложений.Вставить(

ПрисоединенныйФайл.Наименование,

Новый Картинка(ПолучитьИзВременногоХранилища(

ПрисоединенныеФайлы.ПолучитьДанныеФайла(ПрисоединенныйФайл.Ссылка).СсылкаНаДвоичныеДанныеФайла)));

Правильно разбивать такие вызовы на отдельные операторы с помощью вспомогательных локальных переменных:

АдресФайлаИзображения = ПрисоединенныеФайлы.ПолучитьДанныеФайла(ПрисоединенныйФайл.Ссылка).СсылкаНаДвоичныеДанныеФайла;

ДанныеИзображения = Новый Картинка(ПолучитьИзВременногоХранилища(АдресФайлаИзображения));

СтруктураВложений.Вставить(ПрисоединенныйФайл.Наименование, ДанныеИзображения);

В то же время, если код с вложенными вызовами получается компактным (не требует переноса выражений) и легко читаемым, то вложенные вызовы допустимы.  
Например:

Предупреждение(НСтр("ru='Для выполнения операции необходимо установить расширение работы с файлами.'"));

ПеречитатьСуммуПоКурсу(Сумма, КурсВалютыНаДату(Валюта));

6.2. Также не рекомендуется при вызове функций использовать вложенный конструктор структуры: **Новый Структура(...)**. Вложенное объявление структуры допустимо только в тех случаях, когда количество ее свойств небольшое (нужно ориентироваться на количество свойств не более трех).

Неправильно:

ЗаполнитьЦены(

Объект.Товары, // Табличная часть

, // Массив строк или структура отбора

Новый Структура( // Параметры заполнения

"Дата, Валюта, Соглашение, ПоляЗаполнения",

Объект.Дата,

Объект.Валюта,

Объект.Соглашение,

"Цена, СтавкаНДС, ВидЦены, СрокПоставки"

),

Новый Структура( // Структура действий с измененными строками

"ПересчитатьСумму, ПересчитатьСуммуСНДС, ПересчитатьСуммуНДС, ПересчитатьСуммуРучнойСкидки, ОчиститьАвтоматическуюСкидку, ОчиститьСуммуВзаиморасчетов",

"КоличествоУпаковок", СтруктураПересчетаСуммы, СтруктураПересчетаСуммы, "КоличествоУпаковок", Неопределено, Неопределено

)

);

Правильно:

ПараметрыЗаполнения = Новый Структура;

ПараметрыЗаполнения.Вставить("Дата", Объект.Дата);

ПараметрыЗаполнения.Вставить("Валюта", Объект.Валюта);

ПараметрыЗаполнения.Вставить("Соглашение", Объект.Соглашение);

ПараметрыЗаполнения.Вставить("ПоляЗаполнения", "Цена, СтавкаНДС, ВидЦены, СрокПоставки");

ДействияСИзмененнымиСтроками = Новый Структура;

ДействияСИзмененнымиСтроками.Вставить("ПересчитатьСумму","КоличествоУпаковок");

ДействияСИзмененнымиСтроками.Вставить("ПересчитатьСуммуСНДС", ПараметрыПересчетыСуммы);

ДействияСИзмененнымиСтроками.Вставить("ПересчитатьСуммуНДС", ПараметрыПересчетыСуммы);

ДействияСИзмененнымиСтроками.Вставить("ПересчитатьСуммуРучнойСкидки","КоличествоУпаковок");

ДействияСИзмененнымиСтроками.Вставить("ОчиститьАвтоматическуюСкидку");

ДействияСИзмененнымиСтроками.Вставить("ОчиститьСуммуВзаиморасчетов ");

ЗаполнитьЦены(Объект.Товары, ПараметрыЗаполнения, ДействияСИзмененнымиСтроками);

7. При вызове функций не следует пропускать обязательные параметры. В противном случае, в параметр будет передано значение **Неопределено**, на которое функция может быть не рассчитана. Если же значение **Неопределено** является допустимым, то нужно или его передавать в функцию явно, или сделать этот параметр необязательным со значением по умолчанию **Неопределено**.

Например, для вызова процедуры

Процедура ПоменятьЦветПоляФормы(Форма, ИмяПоля, Цвет)

неправильно:

ПоменятьЦветПоляФормы(,"РезультатПроверки", ЦветаСтиля.ПоясняющийОшибкуТекст); // пропущен первый параметр Форма

ПоменятьЦветПоляФормы(,,); // пропущены все обязательные параметры

правильно:

ПоменятьЦветПоляФормы(ЭтотОбъект, "РезультатПроверки", Цвет); // указаны все обязательные параметры

См. также

* [Передача параметров по ссылке и по значению при вызове процедур и функций](https://its.1c.ua/db/metod81/content/2606/hdoc) (статья на ИТС)
* [Особенности использования структур в качестве параметров процедур и функций](https://its.1c.ua/db/content/v8std/src/300/100/i8100641.htm?_=1580137113)
* [Использование объектов типа Структура](https://its.1c.ua/db/content/v8std/src/300/300/i8100693.htm?_=1580137113)

### Особенности использования структур в качестве параметров процедур и функций

#std641

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Основная статья: [*Параметры процедур и функций*](https://its.1c.ua/db/content/v8std/src/300/100/i8100640.htm?_=1580137113)

Для процедур и функций (далее по тексту: функций) с параметрами типа **Структура** рекомендуется придерживаться следующего подхода к разработке.

1. Помимо функции, которая собственно реализует прикладную функциональность (далее по тексту: вызываемая функция), необходимо определить функцию-конструктор для создания новой структуры (далее по тексту функция-конструктор параметров). При этом сама функция-конструктор не принимает параметров, а только возвращает структуру-заготовку со свойствами, которую вызывающий код должен проинициализировать конкретными значениями и передать в вызываемую функцию.

Пример вызывающего кода:

Процедура ПриИзмененииНоменклатурыСервер(ИдентификаторТекущейСтроки)

// Получаем новую структуру параметров.

ПараметрыЗаполненияЦен = ЦенообразованиеКлиентСервер.ПараметрыЗаполненияЦеныВСтрокеТЧ();

// Заполняем параметры.

ПараметрыЗаполненияЦен.Дата = Объект.Дата;

ПараметрыЗаполненияЦен.Валюта = Объект.Валюта;

ТекущаяСтрока = Объект.Товары.НайтиПоИдентификатору(ИдентификаторТекущейСтроки);

// Передаем структуру параметров в прикладную функцию.

ЦенообразованиеСервер.ЗаполнитьЦеныВСтрокеТЧ(ТекущаяСтрока, ПараметрыЗаполненияЦен);

КонецПроцедуры

Пример функции-конструктора параметров в модуле **ЦенообразованиеКлиентСервер**:

Функция ПараметрыЗаполненияЦеныВСтрокеТЧ() Экспорт

ПараметрыЗаполненияЦен = Новый Структура;

ПараметрыЗаполненияЦен.Вставить("Дата");

ПараметрыЗаполненияЦен.Вставить("Валюта");

ПараметрыЗаполненияЦен.Вставить("ПересчитыватьСумму", Истина);

ПараметрыЗаполненияЦен.Вставить("ОбязательныеПараметры","Дата,Валюта"); // обязательные параметры, которые нужно заполнять

Возврат ПараметрыЗаполненияЦен;

КонецФункции

Имена свойств структуры соответствуют параметрам вызываемой функции. При этом параметры со значениями по умолчанию должны быть явно проинициализированы в этой структуре.

2. В вызывающем коде не следует инициализировать структуру параметров или добавлять в нее какие-либо другие свойства. Во избежание неоднозначности и скрытых ошибок все допустимые параметры вызываемой функции должны быть определены явно в функции-конструкторе параметров.

См. также

* [Использование объектов типа Структура](https://its.1c.ua/db/content/v8std/src/300/300/i8100693.htm?_=1580137113)

### Правила образования имен переменных

#std454

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Имена переменных следует образовывать от терминов предметной области таким образом, чтобы из имени переменной было понятно ее назначение.

2. Имена следует образовывать путем удаления пробелов между словами. При этом, каждое слово в имени пишется с прописной буквы. Предлоги и местоимения из одной буквы также пишутся прописными буквами.  
Пример:

Перем ДиалогРаботыСКаталогом; // Диалог работы с каталогом

Перем КоличествоПачекВКоробке; // Количество пачек в коробке

Примеры некорректных имен переменных:

масРеквизитов, соотвВидИмя, новСтр

3. Имена переменных запрещается начинать с подчеркивания.

4. Имена переменных не должны состоять из одного символа. Использование односимвольных имен переменных допускается только для счетчиков циклов.

5. Переменные, отражающие состояние некоторого флага, следует называть так, как пишется истинное значение этого флага.  
Например:

Перем ЕстьОшибки; // Признак наличия ошибок в процедуре.

Перем ЭтоТоварТара; // Признак, что товар относится к возвратной таре.

См. также

* [Использование глобальных переменных в программных модулях](https://its.1c.ua/db/content/v8std/src/300/200/i8100639.htm?_=1580137113)

### Работа с параметром «Отказ» в обработчиках событий

#std686

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В обработчиках событий модулей объектов, наборов записей, форм и т.п., содержащих параметр **Отказ** (**ПриЗаписи**, **ОбработкаПроверкиЗаполнения**, **ТоварыПередНачаломДобавления** и т.п.), не следует присваивать этому параметру значение **Ложь**.

Это требование обусловлено тем, что, как правило, в коде обработчиков событий параметр **Отказ** может устанавливаться сразу в нескольких последовательных проверках (или в нескольких подписках на одно и то же событие). В таком случае к моменту выполнения очередной проверки параметр **Отказ** уже может заранее содержать значение **Истина**, и можно ошибочно сбросить его обратно в **Ложь**.

Кроме того, при доработках конфигурации на внедрении число этих проверок может увеличиться.

Неправильно:

Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)

...

Отказ = ЕстьОшибкиЗаполнения();

...

КонецПроцедуры

Правильно:

Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)

...

Если ЕстьОшибкиЗаполнения() Тогда

Отказ = Истина;

КонецЕсли;

...

КонецПроцедуры

или

Отказ = Отказ Или ЕстьОшибкиЗаполнения();

2. Эти же требования справедливы для других аналогичных параметров обработчиков событий: **СтандартнаяОбработка**, **Выполнение** и др.  
Например:

Процедура ОбработкаПолученияДанныхВыбора(ДанныеВыбора, Параметры, СтандартнаяОбработка)

Если Параметры.Свойства(...) Тогда

СтандартнаяОбработка = Ложь;

...

КонецЕсли;

КонецПроцедуры

## Использование конструкций встроенного языка

### Общие требования к построению конструкций встроенного языка

#std441

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В конструкциях встроенного языка ключевые слова пишутся канонически (как в документации или Синтакс-помощнике).  
Правильно:

КонецЕсли

Неправильно:

конецЕсли, КОНЕЦЕСЛИ, конецесли, Конецесли.

2. При следовании друг за другом нескольких операторов присваивания, допускается выравнивать их следующим образом:

ДиалогВыбора.ПолноеИмяФайла = ИмяФайла;  
ДиалогВыбора.Каталог        = ИмяПути;  
ДиалогВыбора.Заголовок      = НСтр("ru = 'Выберите файл со списком запросов'");  
ДиалогВыбора.Фильтр         = НСтр("ru = 'Файлы запросов (\*.sel)|\*.sel|Все файлы (\*.\*)|\*.\*'");  
ДиалогВыбора.Расширение     = "sel";

При этом не следует выравнивать операторы одинаково по всему модулю - рекомендуется делать выравнивание только для операторов, расположенных рядом.

3. Составные логические выражения в **Если**…**КонецЕсли** переносятся согласно [правилам переноса выражений](https://its.1c.ua/db/content/v8std/src/300/200/i8100444.htm?_=1580137113).

4. Логические выражения и логические значения (например, результат функции, возвращающей логическое значение, переменные типа **Булево** и пр.) не следует проверять путем сравнения с литералами **Истина** и **Ложь**.  
Правильно:

Если ЭтоНовый() Тогда

Неправильно:

Если ЭтоНовый() = Истина Тогда

5. В тех случаях, когда требуется сравнивать результаты каких-либо выражений, следует предварительно присваивать результаты этих выражений промежуточным переменным, и сравнивать уже сами эти переменные.  
Правильно:

Ответ = Вопрос(НСтр("ru = 'Данные еще не записаны. Записать?'"), РежимДиалогаВопрос.ДаНет,, КодВозвратаДиалога.Да);  
Если Ответ = КодВозвратаДиалога.Да Тогда  
  Записать();  
Иначе  
  Возврат;  
КонецЕсли;

Неправильно:

Если Вопрос(НСтр("ru = 'Данные еще не записаны. Записать?'"), РежимДиалогаВопрос.ДаНет,, КодВозвратаДиалога.Да) = КодВозвратаДиалога.Да Тогда  
  Записать();  
Иначе  
  Возврат;  
КонецЕсли;

6. Необходимо использовать системные наборы значений везде, где возможно их применить, например, вместо **Символ(10)** следует использовать **Символы.ПС**.

См. также

* [Перенос выражений](https://its.1c.ua/db/content/v8std/src/300/200/i8100444.htm?_=1580137113)
* [Тексты модулей](https://its.1c.ua/db/content/v8std/src/300/100/i8100456.htm?_=1580137113)

### Перенос выражений

#std444

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При длине строки более 120 символов следует использовать переносы. Строки длиннее 120 символов делать не рекомендуется, за исключением тех случаев, когда перенос невозможен.

2. Длинные арифметические выражения переносятся следующим образом:

* в одной строке может находиться более одного операнда;
* при переносе знаки операции пишутся в начале строки (а не в конце предыдущей строки);
* операнды на новой строке предваряются стандартным отступом, либо выравниваются по началу первого операнда без учета знаков операций.

Пример:

СуммаДокумента = СуммаБезСкидки  
                 + СуммаРучнойСкидки  
                 + СуммаАвтоматическойСкидки;

или

СуммаДокумента = СуммаБезСкидки  
    + СуммаРучнойСкидки  
    + СуммаАвтоматическойСкидки;

3.1 Длинные строковые константы рекомендуется переносить с помощью специального символа перевода на новую строку, например:

  ТекстЗапроса =  
  "ВЫБРАТЬ РАЗРЕШЕННЫЕ  
  | ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок  
  |ИЗ  
  | РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету  
  |ГДЕ  
  | ЗаметкиПоПредмету.Предмет = &Предмет";

или

ТекстПредупреждения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(  
  НСтр("ru = 'Обновление адресного классификатора не требуется.  
             |В программе уже загружены актуальные адресные сведения от %1.'"),  
  Формат(ДатаПоследнегоОбновленияКЛАДР, "ДЛФ=D"));  
ПоказатьПредупреждение(,ТекстПредупреждения);

При этом не следует переносить строки, содержащие текст сообщения пользователю (объект **СообщениеПользователю**).

3.2. В общем случае при конкатенации строк знак "+" рекомендуется писать в начале строки, так же как и при переносе арифметических выражений (см. п.2), например:

ПоляОтбора = "Номенклатура,Характеристика,Склад"  
   + ДополнительныеПоляОтбора;

3.3. При конкатенации длинных строк знак "+" можно писать в конце строки, чтобы не ломать общее форматирование текста. Например,

ТекстЗапроса = ТекстЗапроса +  
"ВЫБРАТЬ  
| Номенклатура.Ссылка КАК Ссылка  
|ИЗ  
| Справочник. Номенклатура КАК Номенклатура";

4. При необходимости параметры процедур, функций и методов следует переносить следующим образом:

* параметры выравниваются по началу первого параметра, либо предваряются стандартным отступом;
* закрывающая скобка и разделитель операторов ";" пишутся в той же строке, что и последний параметр;
* также допустим и способ форматирования, который предлагает функция автоформатирования в конфигураторе (см. п. 5).

Пример:

ИменаДокументов = Новый СписокЗначений;  
ИменаДокументов.Добавить(Метаданные.Документы.СтрокаВыпискиРасход.Имя,   
                         Метаданные.Документы.СтрокаВыпискиРасход.Синоним);  
ИменаДокументов.Добавить(Метаданные.Документы.РасходныйКассовыйОрдер.Имя,   
                         Метаданные.Документы.РасходныйКассовыйОрдер.Синоним);

или

ИменаДокументов = Новый СписокЗначений;  
ИменаДокументов.Добавить(Метаданные.Документы.СтрокаВыпискиРасход.Имя,   
    Метаданные.Документы.СтрокаВыпискиРасход.Синоним);  
ИменаДокументов.Добавить(Метаданные.Документы.РасходныйКассовыйОрдер.Имя,   
    Метаданные.Документы.РасходныйКассовыйОрдер.Синоним);

5. Сложные логические условия в **Если…ИначеЕсли…КонецЕсли** следует переносить следующим образом:

* каждое элементарное условие нужно начинать с новой строки, если длина строки превышает ограничение в 120 символов;
* логические операторы **И**, **ИЛИ** ставятся в начале строки, а не в конце предыдущей строки;
* все условия предваряются стандартным отступом, либо выравниваются по началу первого условия, без учета логического оператора (для выравнивания выражений относительно первой строки рекомендуется использовать пробелы).

Примеры:

Если (ВидОперации = Перечисления.ВидыОперацийПоступлениеМПЗ.ПоступлениеРозница)  
  ИЛИ (ВидОперации = Перечисления.ВидыОперацийПоступлениеМПЗ.ПоступлениеРозницаКомиссия) Тогда  
  Возврат Истина;  
КонецЕсли;

Если ((СтруктураМодуля[Индекс].Блок = Перечисления.ТипыБлоковМодулей.ЗаголовокПроцедуры)  
  ИЛИ(СтруктураМодуля[Индекс].Блок = Перечисления.ТипыБлоковМодулей.ЗаголовокФункции))  
  И(Найти(ВРЕГ(СтруктураМодуля[Индекс].Текст), КлючБлока)> 0) Тогда

6. Для выполнения перечисленных выше рекомендаций, кроме автоматического форматирования текста программного модуля, в процессе ввода можно также отформатировать уже введенный текст. Для этого необходимо выделить блок текста, который требуется отформатировать, и выбрать пункт меню **Текст — Блок — Форматировать**. При этом текстовый редактор проанализирует текст модуля и выполнит его форматирование, при котором содержимое каждой синтаксической конструкции будет сдвинуто вправо на величину табуляции независимо от первоначального расположения строк (лидирующих пробелов). В пустые строки устанавливаются знаки табуляции в соответствии с синтаксической конструкцией.

Для автоматической расстановки переносов строк можно воспользоваться [приложенной обработкой](https://its.1c.ua/db/files/1CITS/EXE/V8Std/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D0%9F%D0%B5%D1%80%D0%B5%D0%BD%D0%BE%D1%81%D0%92%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D0%9F%D0%B5%D1%80%D0%B5%D0%BD%D0%BE%D1%81%D0%92%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9.zip).

### Использование дублирующего кода

#std440

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Дублированием кода называется способ разработки конфигурации, при котором при создании нового функционала копируются без изменений уже существующие фрагменты кода или целиком процедуры и функции и при этом копируемый функционал по логике приложения должен быть одинаковым.

Дублирование кода создает проблемы для сопровождения разрабатываемой конфигурации:

* в копию попадают все ошибки из дублируемого кода;
* при исправлении ошибок существует вероятность пропустить некоторые вхождения;
* затраты на исправление ошибок увеличиваются;
* усложняется понимание структуры программы.

Дублирование часто возникает из-за невозможности доступа к написанному прежде коду (например, если написанный прежде код размещен в модуле той или иной формы, а его использование востребовано при разработке другой формы).  
  
Следует осторожно относиться к дублированию кода и по возможности стараться его избегать. Основной способ избежать дублирования кода - переработать существующий код. Это позволит вывести процедуры и функции, алгоритмы которых могут быть использованы повторно, из модулей объектов и модулей форм в общие модули.

2. Следует помнить, что дублирование кода оправданно и должно выполняться, если развитие функционала в будущем может привести к значительному расхождению двух вариантов кода.

#### Пример предотвращения дублирования кода при разработке клиент-серверных функций

В функции **СообщитьПользователю** общего модуля **ОбщегоНазначенияКлиентСервер** возникла необходимость особым образом обработать входные параметры при выполнении на сервере.

Неправильно использовать для этого [инструкции препроцессора](https://its.1c.ua/db/content/v8std/src/300/200/i8100439.htm?_=1580137113) (#Если НЕ ТонкийКлиент И НЕ ВебКлиент):

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь) Экспорт  
   
 Сообщение = Новый СообщениеПользователю;  
 Сообщение.Текст = ТекстСообщенияПользователю;  
 Сообщение.Поле = Поле;  
   
 ЭтоОбъект = Ложь;  
   
#Если НЕ ТонкийКлиент И НЕ ВебКлиент Тогда  
 Если КлючДанных <> Неопределено  
    И XMLТипЗнч(КлючДанных) <> Неопределено Тогда  
  ТипЗначенияСтрокой = XMLТипЗнч(КлючДанных).ИмяТипа;  
  ЭтоОбъект = СтрНайти(ТипЗначенияСтрокой, "Object.") > 0;  
 КонецЕсли;  
#КонецЕсли  
   
 Если ЭтоОбъект Тогда  
  Сообщение.УстановитьДанные(КлючДанных);  
 Иначе  
  Сообщение.КлючДанных = КлючДанных;  
 КонецЕсли;  
   
 Сообщение.Сообщить();  
 Отказ = Истина;  
   
КонецПроцедуры

Правильно разделить процедуру на две одноименные в серверном и клиентском модуле с различной реализацией, и для того чтобы избежать дублирования кода, общую реализацию оставить в клиент-серверном общем модуле:

1) серверная функция:

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь) Экспорт  
   
 ЭтоОбъект = Ложь;  
 Если КлючДанных <> Неопределено  
  И XMLТипЗнч(КлючДанных) <> Неопределено Тогда  
    
  ТипЗначенияСтрокой = XMLТипЗнч(КлючДанных).ИмяТипа;  
  ЭтоОбъект = СтрНайти(ТипЗначенияСтрокой, "Object.") > 0;  
 КонецЕсли;  
   
 ОбщегоНазначенияСлужебныйКлиентСервер.СообщитьПользователю(ТекстСообщенияПользователю, КлючДанных, Поле, Отказ, ЭтоОбъект);  
   
КонецПроцедуры

2) клиентская функция:

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь) Экспорт  
   
 ОбщегоНазначенияСлужебныйКлиентСервер.СообщитьПользователю(ТекстСообщенияПользователю, КлючДанных, Поле, Отказ);  
   
КонецПроцедуры

3) общая служебная клиент-серверная реализация в модуле ОбщегоНазначенияСлужебныйКлиентСервер:

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных, Знач Поле, Отказ = Ложь, ЭтоОбъект = Ложь) Экспорт  
   
 Сообщение = Новый СообщениеПользователю;  
 Сообщение.Текст = ТекстСообщенияПользователю;  
 Сообщение.Поле = Поле;  
   
 Если ЭтоОбъект Тогда  
  Сообщение.УстановитьДанные(КлючДанных);  
 Иначе  
  Сообщение.КлючДанных = КлючДанных;  
 КонецЕсли;  
   
 Сообщение.Сообщить();  
 Отказ = Истина;  
   
КонецПроцедуры

См. также

* [Разработка конфигураций с повторным использованием общего кода и объектов метаданных](https://its.1c.ua/db/content/v8std/src/800/i8100551.htm?_=1580137113)

### Использование директив компиляции и инструкций препроцессора

#std439

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Директивы компиляции:

&НаКлиенте (&AtClient)  
&НаСервере (&AtServer)  
&НаСервереБезКонтекста (&AtServerNoContext)

следует применять только в коде модулей управляемых форм и в коде модулей команд. В остальных модулях рекомендуется применять инструкции препроцессору.

В серверных или клиентских общих модулях контекст исполнения очевиден, поэтому смысла в директивах компиляции нет. В общих модулях с признаками клиент и сервер применение директив компиляции затрудняет понимание, какие же процедуры (функции) доступны в конечном итоге.

2. Не следует использовать инструкции препроцессора в [клиент-серверных общих модулях](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1580137113) для проверки клиентского и серверного контекстов (#Если Сервер, #Если Клиент) ввиду невозможности надежного определения контекста исполнения. Процедуры и функции, которые работают по-разному при вызове с клиента и с сервера, следует размещать в общих модулях с постфиксами **Клиент** и **Сервер**, а не **КлиентСервер**.

В противном случае невозможно гарантировать корректную работу клиент-серверных процедур и функций в различных режимах работы платформы 1С:Предприятие.

Например, неправильно:

Функция КодОсновногоЯзыка() Экспорт  
#Если НЕ ТонкийКлиент И НЕ ВебКлиент Тогда  
 Возврат Метаданные.ОсновнойЯзык.КодЯзыка;  
#Иначе  
 Возврат СтандартныеПодсистемыКлиент.ПараметрКлиента("КодОсновногоЯзыка");  
#КонецЕсли  
КонецФункции

также неправильно:

Функция КодОсновногоЯзыка() Экспорт  
#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда  
 Возврат Метаданные.ОсновнойЯзык.КодЯзыка;  
#Иначе  
 Возврат СтандартныеПодсистемыКлиент.ПараметрКлиента("КодОсновногоЯзыка");  
#КонецЕсли  
КонецФункции

Правильно: разделить на две одноименные функции в серверном и клиентском модуле с различной реализацией. В общем случае, когда у них имеется определенная общая часть, одинаковая для клиента и сервера, то для того чтобы [избежать дублирования кода,](https://its.1c.ua/db/content/v8std/src/300/200/i8100440.htm?_=1580137113) этот общий код (и только его) следует оставить в клиент-серверном общем модуле и вызывать его из клиентской и серверной функций, соответственно. Тем самым надежно достигается различное поведение в клиентском и серверном контекстах без использования инструкций препроцессора.

В то же время, как и в обычных клиентских модулях, допустимо ветвление кода для учета специфики различных режимов работы клиентского приложения: веб-клиент, тонкий или толстый клиент (например, #Если ВебКлиент).

3. Не следует разрывать инструкциями препроцессора и областями отдельные грамматические конструкции, выражения, а также объявления и места вызова процедур и функций.

Например, неправильно:

Процедура Пример1()  
  а = 1  
#Область ИмяОбласти  
    + 2;  
#КонецОбласти // разрыв выражения  
КонецПроцедуры

#Область ИмяОбласти  
Процедура Пример2()  
    // ...  
#КонецОбласти // разрыв процедуры  
КонецПроцедуры

Если <...> Тогда  
    // ...  
#Если ВебКлиент Тогда // разрыв блока Если  
Иначе  
    // ...  
#КонецЕсли  
КонецЕсли;

Результат = Пример4(Параметр1,  
#Если Клиент Тогда  
  Параметр2, // некорректный вызов функции  
#КонецЕсли  
  Параметр3);

Данные ошибки диагностируются автоматически с помощью среды разработки [1C:Enterprise Development Tools](http://v8.1c.ru/overview/release_IDE_beta/) (EDT).

Правильно использовать инструкции препроцессора без разрыва конструкций.

### Определение типа значения переменной

#std442

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Определение типа значения переменной необходимо выполнять путем его сравнения с типом, а не каким-либо другим методом.  
Правильно:

Если ТипЗнч(Ссылка) = Тип("ДокументСсылка.ПоступлениеТоваровУслуг") Тогда

Неправильно:

Если Ссылка.Метаданные().Имя = "ПоступлениеТоваровУслуг" Тогда

### Получение метаданных объектов

#std445

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В тех случаях, когда известен тип объекта метаданного (справочник, документ, и т.п.), то получение метаданных объекта конфигурации следует выполнять с помощью метода **Метаданные** этого объекта (или ссылки для объектов ссылочного типа), а не путем обращения к свойству глобального контекста **Метаданные**, так как второй способ существенно более медленный.  
  
Правильно:

СправочникОбъект.Метаданные()

Неправильно:

Метаданные.Справочники[ИмяСправочника]  
Метаданные.НайтиПоПолномуИмени("Справочник." + ИмяСправочника)

2. В тех случаях, когда тип объекта метаданного заранее неизвестен, рекомендуется воспользоваться методом **НайтиПоТипу**, например:

// Получить полное имя объекта метаданных  вида "Справочник.Номенклатура", "Документ.ПриходнаяНакладная" по переданной ссылке.  
ИмяОбъектаМетаданного = Метаданные.НайтиПоТипу(ТипЗнч(Ссылка)).ПолноеИмя();

### Обработчики событий модуля формы, подключаемые из кода

#std492

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Обработчикам событий модуля формы, которые устанавливаются из кода с помощью метода **УстановитьДействие**, рекомендуется задавать префикс **Подключаемый\_**(англ. **Attachable\_**). Например:

Процедура Подключаемый\_РазрешитьРедактированиеРеквизитовОбъекта()  
…  
Процедура Подключаемый\_КонтактнаяИнформацияНачалоВыбора()  
…

В случае когда подключение обработчика выполняется не в тексте модуля формы (а например, в общем модуле), то в результатах проверки конфигурации с включенным флажком "Поиск неиспользуемых процедур и функций" окажутся ошибки вида:

Справочник.\_ДемоПартнеры.Форма.ФормаЭлемента.Форма Не обнаружено ссылок на процедуру: " Подключаемый\_КонтактнаяИнформацияНачалоВыбора"

Использование префикса позволяет легко идентифицировать такие обработчики в результатах проверки и отсеивать как исключения.

Если же подключение обработчика выполняется в тексте модуля формы, то проверка конфигурации с включенным флажком "Поиск неиспользуемых процедур и функций" ошибку не регистрирует.

### Использование переменных в программных модулях

#std639

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В большинстве случаев, вместо переменных программных модулей следует использовать более подходящие средства разработки платформы **1С:Предприятие**. Поскольку область видимости (использования) таких переменных сложно контролировать, то они зачастую становятся источником трудновоспроизводимых ошибок.

Примеры некорректного использования и исключений из этого правила приведены далее. Рекомендации по оформлению переменных в коде программных модулей см. в статье [Структура модуля](https://its.1c.ua/db/content/v8std/src/300/100/i8100455.htm?_=1580137113).

2. Неоправданные примеры использования переменных в модулях объектов (справочников, документов, наборов записей, обработок, отчетов и пр.).

2.1. Для передачи параметров между обработчиками подписок на события и в обработчики событий модуля объекта из внешнего кода рекомендуется использовать свойство объекта **ДополнительныеСвойства**. Например, неправильно:

Перем КонвертацияФайлов Экспорт;

Процедура ПередЗаписью(Отказ)  
  
  Если КонвертацияФайлов Тогда   
  ...  
  
КонецПроцедуры

// вызывающий код  
ФайлОбъект.КонвертацияФайлов = Истина;  
ФайлОбъект.Записать();

Правильно:

Процедура ПередЗаписью(Отказ)  
  
  Если ДополнительныеСвойства.Свойство("КонвертацияФайлов") Тогда   
  ...  
  
КонецПроцедуры

// вызывающий код  
ФайлОбъект.ДополнительныеСвойства.Вставить("КонвертацияФайлов", Истина);  
ФайлОбъект.Записать();

В то же время, для передачи внутренних параметров между обработчиками событий модуля объекта целесообразно использовать неэкспортные переменные модуля объекта, которые недоступны из внешнего кода.  
Например:

Перем ПредыдущееЗначениеОрганизации; // значение реквизита "Организация" до записи объекта в базу

Процедура ПередЗаписью(Отказ)  
  ПредыдущееЗначениеОрганизации = ...; // с помощью запроса выясняем значение до записи объекта в базу  
КонецПроцедуры

Процедура ПриЗаписи(Отказ)  
  Если ПредыдущееЗначениеРеквизита <> Организация Тогда  
    // отрабатываем изменение значения реквизита при записи  
    ...  
  КонецЕсли;

КонецПроцедуры

2.2. Для обработки кодов возврата (ошибок) в логике программного модуля рекомендуется использовать строковые константы.  
Например, неправильно:

Перем НетОшибок,  
Ошибка\_ОбработкиПроверкиЗаполнения, // возникает, если обработка проверки заполнения вернула отказ  
Ошибка\_ЗаписиОбъекта, // возникает, если во время записи объекта возникло исключение  
Ошибка\_БлокировкиОбъекта, // возникает, при попытке блокировки объекта

Процедура ВыполнитьПерерасчет()  
  ...  
  Результат = ОбработатьДокументы(...);  
  Если Результат = Ошибка\_ЗаписиОбъекта Тогда  
    ...  
  ИначеЕсли Результат = Ошибка\_БлокировкиОбъекта Тогда  
    ...  
  ИначеЕсли ...

КонецПроцедуры

...

////////////////////////////////////////////////////////////////////////////////  
// ИНИЦИАЛИЗАЦИЯ МОДУЛЯ

НетОшибок     = 1;  
Ошибка\_ОбработкиПроверкиЗаполнения = 2;  
Ошибка\_ЗаписиОбъекта    = 3;  
Ошибка\_БлокировкиОбъекта   = 4;

правильно:

Процедура ВыполнитьПерерасчет()  
  ...  
  Результат = ОбработатьДокументы(...);  
  Если Результат = "ОшибкаЗаписиОбъекта" Тогда  
    ...  
  ИначеЕсли Результат = "ОшибкаБлокировкиОбъекта" Тогда  
    ...  
  ИначеЕсли ...

КонецПроцедуры

2.3. Для кеширования долго-вычисляемых и часто-используемых значений в процедурах и функциях рекомендуется применять [модули с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100459.htm?_=1580137113) на время вызова сервера.

Исключение из этого правила составляют случаи, когда по соображениям безопасности возвращать результат вычисления в экспортной функции недопустимо. В этом случае они размещаются в локальной переменной модуля.

3. Неоправданные примеры использования переменных в модулях форм.

3.1. Для кеширования долго-вычисляемых и часто-используемых значений в процедурах и функциях рекомендуется применять [модули с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100459.htm?_=1580137113).

При этом не следует кешировать статическую и легко вычисляемую информацию. В частности, не следует кешировать в клиентских переменных модуля формы значения предопределенных элементов и перечислений. Для их получения на клиенте предназначен [метод **ПредопределенноеЗначение**](https://its.1c.ua/db/content/v8std/src/400/i8100443.htm?_=1580137113).

3.2. Для хранения и передачи промежуточных результатов вычислений между разными процедурами и функциями формы следует использовать

* [Параметры процедур и функций](https://its.1c.ua/db/content/v8std/src/300/100/i8100640.htm?_=1580137113) – для передачи результатов по цепочке вызовов дочерних процедур и функций в контексте одного вызова.
* Реквизиты формы – если требуется сохранять промежуточные результаты между разными вызовами с клиента. (Следует иметь в виду, что значения серверных переменных модуля формы не сохраняются между вызовами с клиента.)

Исключение из этого правила составляют случаи использования клиентских переменных формы для хранения промежуточных результатов в обработчиках ожидания формы, в обработчиках внешних событий и в клиентских обработчиках событий элементов формы.  
Например:

&НаКлиенте  
Перем ПорядковыйНомерИзображения; // счетчик-нумератор для наименования файлов при сканирования нескольких изображений  
...  
&НаКлиенте  
Процедура ВнешнееСобытие(Источник, Событие, Данные)  
 Если Источник = "TWAIN" И Событие = "ImageAcquired" Тогда  
  Если ПорядковыйНомерИзображения = Неопределено Тогда  
   ПорядковыйНомерИзображения = 1;  
  КонецЕсли;   
  ПорядковыйНомерИзображения = ПорядковыйНомерИзображения + 1;  
  // Сохранение отсканированного документа в файл с номером ПорядковыйНомерИзображения  
  // ...  
 КонецЕсли;   
КонецПроцедуры

4. Переменные управляемого и обычного приложения следует использовать для хранения «клиентских параметров сеанса». Подробнее см. статью [Использование параметров сеанса](https://its.1c.ua/db/content/v8std/src/100/100/i8100413.htm?_=1580137113).

### Предварительная инициализация локальных переменных

#std494

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  В случаях когда фрагмент кода вычисляет значение одной или нескольких локальных переменных, рекомендуется явно выполнять предварительную инициализацию таких переменных. Это позволит избежать потенциальных ошибок времени выполнения, когда  значение переменной оказывается **Неопределено**, а последующий код рассчитывает на определенный тип значения. Например:  Если ЧтоТоТам Тогда       МояПеременная = 10; ИначеЕсли       // ряд следующих веток       … КонецЕсли; ... = МояПеременная;  // если ЧтоТоТам не ИСТИНА, то нужно учитывать, что МояПеременная может быть равна Неопределено  Правильно:  МояПеременная = 0; // значение по умолчанию Если ЧтоТоТам Тогда       МояПеременная = 10; ИначеЕсли       // ряд следующих веток       … КонецЕсли;  ... = МояПеременная;  // используем значение переменной, которая всегда имеет числовое значение  Данная рекомендация имеет смысл для больших блоков Если/ИначеЕсли/Иначе, внутри которых сложно визуально по тексту контролировать инициализацию переменных. |

### Использование Журнала регистрации

#std498

Область применения: управляемое приложение, обычное приложение.

1. Журнал регистрации предназначен для хранения событий, возникающих в процессе работы пользователей с информационной базой. При администрировании эту информацию часто необходимо анализировать в различных разрезах для того, чтобы например, узнать какие события происходили в определенный момент времени, какие действия выполнял тот или иной пользователь.

2. Рекомендуется производить запись в Журнал регистрации из встроенного языка в тех случаях, когда администратору необходимо сообщить дополнительную диагностическую информацию о событиях, которые не записываются платформой **1С:Предприятие**. Такая необходимость может возникнуть как при выполнении бизнес-логики, вызываемой при интерактивной работе, так и в фоновых (регламентных) заданиях. Для удобства анализа Журнала регистрации одна его запись должна соответствовать одному событию, а сами записи должны содержать ряд обязательных атрибутов, в разрезе которых проводится анализ.

2.1.**Строковый идентификатор типа события.** Как правило, список типов событий в конфигурации может быть сколь угодно большим, поэтому типы событий рекомендуется группировать по функциональному признаку: «Название группы событий.Название события». Например, правильно записывать события с типами «Поручения.Уведомление о новых задачах» и «Поручения.Уведомление о зависших задачах» вместо двух «плоских» типов событий «Уведомление о новых задачах» и «Уведомление о зависших задачах». Текст типа события – локализуем, при этом всегда задается основной язык конфигурации.

2.2.**Уровень важности события.** Критичные события, требующие повышенного внимания администратора (ошибки бизнес-логики, сбои в программе, и т.п.), записываются в Журнал регистрации с уровнем важности «Ошибка». Потенциальные проблемы и не фатальные ошибки регистрируются как «Предупреждения». Для вывода информационных сообщений об успешном завершении той или иной операции используется уровень важности «Информация». Также возможно применять и более низкий уровень важности – «Примечание».

2.3. **Комментарий.** Содержит текстовую неструктурированную информацию о событии. В случае ошибок в этом поле содержится информация, необходимая для расследования причины проблемы. Не следует помещать в комментарий информацию сразу о нескольких событиях. Например, неправильно записывать одно событие с комментарием вида:

[01.01.2010 00:00:01] Начало инициализации обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1  
[01.01.2010 00:00:02] Окончание инициализации обмена данными (успешно)  
[01.01.2010 00:00:03] Начало процесса обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1  
[01.01.2010 00:00:04] Начало записи изменений в файл обмена  
[01.01.2010 00:00:05] Окончание записи изменений в файл обмена (успешно)  
[01.01.2010 00:00:06] Окончание процесса обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1  
[01.01.2010 00:00:07] Выполнено, Выгрузка данных, Обработано 1 объектов

правильно записать столько событий, сколько их реально произошло.

Текст комментария – локализуем. Для записи в Журнал регистрации информации о возникшем исключении следует использовать конструкцию:

ПодробноеПредставлениеОшибки(ИнформацияОбОшибке())

Пример регистрации дополнительных событий в функциональной подсистеме «Мой механизм»:

Попытка   
 ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоЯзыка),  
  УровеньЖурналаРегистрации.Информация, , ,  
  НСтр("ru = 'Начато действие'"));  
 ДействиеСВозможнойОшибкой(ОбъектДействия);  
 ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоЯзыка),  
  УровеньЖурналаРегистрации.Информация, , ,  
  НСтр("ru = 'Завершено действие'");  
Исключение  
 ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоЯзыка),  
  УровеньЖурналаРегистрации.Ошибка, , ,  
  НСтр("ru = '"Во время выполнения действия произошла неизвестная ошибка.'") + Символы.ПС +  
  ПодробноеПредставлениеОшибки(ИнформацияОбОшибке());  
 КонецПопытки;    
КонецПроцедуры

где переменная **КодОсновногоЯзыка** содержит код языка для хранения данных в информационной базе. Подробнее см. [Автогенерированные данные в информационной базе: требования по локализации](https://its.1c.ua/db/content/v8std/src/900/i8100784.htm?_=1580137113), п. 1.

3. Не следует использовать выборку из журнала регистрации в тех задачах, где критична высокая скорость выполнения выборки. Поскольку при больших объемах журнала регистрации скорость выборки падает пропорционально увеличению его объема.

Рекомендуется заводить отдельный регистр для протоколирования интересующих событий или обращаться к специализированным объектам платформы (например, **МенеджерФоновыхЗаданий** для выборки истории выполнения фоновых заданий).

Эту особенность нужно также учитывать при разработке отчетов по журналу регистрации.

См. также

* [Строки интерфейса в модулях конфигурации: требования по локализации](https://its.1c.ua/db/content/v8std/src/900/i8100761.htm?_=1580137113)

### Перехват исключений в коде

#std499

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. В общем случае не рекомендуется перехватывать исключения. В частности не нужно перехватывать исключения только ради выдачи сообщения об ошибке. Необработанное исключение в любом случае будет выдано пользователю в виде сообщения об ошибке (а также будет записано в журнал регистрации для администратора, если исключение возникло на сервере).

2. Тем не менее, необходимость перехвата исключений в коде все же возникает. Например, для того чтобы уточнить текст ошибки, дополнив его прикладной, понятной конечному пользователю, информацией. Однако при этом необходимо фиксировать причину ошибки в журнале регистрации для того, чтобы системный администратор имел возможность выполнить диагностику проблемы и при необходимости передать информацию об ошибке в службу технической поддержки.

При этом рекомендуется записывать в журнал регистрации подробное представление исключения, а краткое представление добавлять в текст сообщения пользователю.

3. Частные случаи некорректного использования и перехвата исключений.

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  3.1. Если имеется некоторая серверная бизнес-логика, которая вызывается с клиента при интерактивной работе пользователя:  &НаСервере Процедура ВыполнитьОперацию()     // код, приводящий к вызову исключения     .... КонецПроцедуры  то неправильно маскировать от пользователя и администратора исходную проблему:  // на клиенте Попытка     ВыполнитьОперацию(); Исключение     ПоказатьПредупреждение(,НСтр("ru = 'Операция не может быть выполнена.'")); КонецПопытки;  Правильно записывать в журнал регистрации подробное представление исключения, а краткое представление добавлять в текст сообщения пользователю:  &НаСервере Процедура ВыполнитьОперацию()   Попытка     // код, приводящий к вызову исключения     ....   Исключение     // Запись события в журнал регистрации для системного администратора.     ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),        УровеньЖурналаРегистрации.Ошибка,,,        ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));     ВызватьИсключение;   КонецПопытки; КонецПроцедуры  и тогда на клиенте:  Попытка     ВыполнитьОперацию(); Исключение     ТекстСообщения = КраткоеПредставлениеОшибки(ИнформацияОбОшибке());     ПоказатьПредупреждение(,НСтр("ru = 'Операция не может быть выполнена по причине:'") + Символы.ПС + ТекстСообщения); КонецПопытки;  3.2. Не следует анализировать текст исключений с целью интерпретации причины ошибки. Текст исключения может [меняться в зависимости от локализации](https://its.1c.ua/db/content/v8std/src/900/i8100761.htm?_=1580137113). В условиях отсутствия штатных средств (например, типизированных исключений), следует выдавать пользователю тексты исключений «как есть». Для понятности, его можно дополнить пояснением возможных причин. Например:  Попытка     ЗагрузитьФайлИзИнтернета(...); Исключение     ТекстСообщения = КраткоеПредставлениеОшибки(ИнформацияОбОшибке());     ТекстСообщения = НСтр("ru = 'Не удалось загрузить файл:'") + Символы.ПС + ТекстСообщения + Символы.ПС + НСтр("ru = 'Возможные причины: • Нет подключения к Интернету; • На веб-узле возникли неполадки; • Брандмауэр или другое промежуточное ПО (антивирусы и т.п.) блокируют попытки программы подключиться к Интернету; • Подключение к Интернету выполняется через прокси-сервер, но его параметры не заданы в программе.'");     ПоказатьПредупреждение(,ТекстСообщения); КонецПопытки;  В тех случаях, когда анализ типов исключений критически важен для корректной работы бизнес-логики, следует отказаться от исключений и использовать коды ошибок (коды возврата). При этом недопустимо использовать числовые коды ошибок, т.к. код становится нечитаемым:  КодОшибки = ЗагрузитьФайлИзИнтернета(...); Если КодОшибки = 12345 Тогда ... ИначеЕсли ...  правильно применять строковые литералы (например, "Успешно", "НетМестаНаДиске", "Отменено" и т.п.):  РезультатЗагрузки = ЗагрузитьФайлИзИнтернета(...); Если РезультатЗагрузки = "Успешно" Тогда ... ИначеЕсли ...  Строковые литералы кодов ошибок не [локализуются](https://its.1c.ua/db/content/v8std/src/900/i8100764.htm?_=1580137113).  Исключение составляют случаи работы с веб-сервисами и другими внешними системами, где коды ошибок не доступны, а результат работы транслируется в вызывающий код прикладной конфигурации в виде исключений.  3.3. Если имеется некоторая клиентская бизнес-логика (код выполняется полностью на клиенте):  &НаКлиенте Процедура СоздатьФайлНаДиске()     // код, приводящий к вызову исключения     .... КонецПроцедуры  то рекомендуется делать дополнительный серверный вызов для протоколирования неудачного результата операции в журнал регистрации:  Попытка     // клиентский код, приводящий к вызову исключения     СоздатьФайлНаДиске(); Исключение     ТекстСообщения = КраткоеПредставлениеОшибки(ИнформацияОбОшибке());     ПоказатьПредупреждение(,НСтр("ru = 'Операция не может быть выполнена по причине:'") + Символы.ПС + ТекстСообщения);     ЗаписатьОшибкуРаботыСФайлами(ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()))); КонецПопытки;  &НаСервереБезКонтекста Процедура ЗаписатьОшибкуРаботыСФайлами(...)     ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),        УровеньЖурналаРегистрации.Ошибка,,,        ПодробноеПредставлениеОшибки); КонецПроцедуры  3.4. Недопустимо перехватывать любые исключения, бесследно для системного администратора:  Попытка     // код, приводящий к вызову исключения     .... Исключение // перехват любых исключений КонецПопытки;  Как правило, подобная конструкция скрывает реальную проблему, которую впоследствии невозможно диагностировать. Правильно:  Попытка     // код, приводящий к вызову исключения     .... Исключение     // Пояснение причин перехвата всех исключений "незаметно" от пользователя.     // ....     // И запись события в журнал регистрации для системного администратора.     ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),        УровеньЖурналаРегистрации.Ошибка,,,        ПодробноеПредставлениеОшибки(ИнформацияОбОшибке())); КонецПопытки;  См. также [Доступ к файловой системе из кода конфигурации](https://its.1c.ua/db/content/v8std/src/400/i8100542.htm?_=1580137113), удаление временных файлов |

3.5. Недопустимо делать проверки наличия у объекта реквизитов, методов, макетов и т.п., используя для этого исключения, т.к. это может привести к сложно диагностируемым ошибкам, а также затрудняет отладку в режиме «Останавливаться по ошибке».  
Вместо перехвата исключений в этом случае рекомендуется:

* использовать механизмы работы с метаданными, чтобы явным образом проверять наличие или отсутствие реквизита (макета и т.п.);
* если различия связаны с особенностями встраивания библиотек – описывать особенности явным образом в переопределяемых модулях (см. [Переопределяемые и поставляемые объекты](https://its.1c.ua/db/content/v8std/src/800/i8100553.htm?_=1580137113));
* пересмотреть логику работы методов, использующих перехват исключений. Например, можно предусмотреть параметры, которые определяются в вызывающем коде и указывают нужно или нет обращаться к какому-либо методу или свойству объекта.

Например, неправильно:

Попытка  
 КонтекстЭДОСервер.ПолучитьМакет("КомпонентаОбмена");  
 ПутьВК = КонтекстЭДОСервер.ПутьКОбъекту +  ".Макет.КомпонентаОбмена";  
Исключение  
КонецПопытки;

Правильно:

МакетКомпонентыОбмена = КонтекстЭДОСервер.Метаданные().Макеты.Найти("КомпонентаОбмена");  
Если МакетКомпонентыОбмена <> Неопределено Тогда  
 ПутьКМакету = КомпонентаОбмена.ПолноеИмя()  
КонецЕсли;

3.6. Порядок обработки исключений при использовании транзакций описан в стандарте [Транзакции: правила использования](https://its.1c.ua/db/content/v8std/src/200/300/i8100783.htm?_=1580137113).

3.7. Неправильно использовать исключения для приведения значения к типу. Для таких операций необходимо использовать возможности объекта **ОписаниеТипов**.

Например, неправильно:

Попытка  
 КоличествоДнейРазрешения = Число(Значение);  
Исключение  
 КоличествоДнейРазрешения = 0; // значение по умолчанию  
КонецПопытки;

Правильно:

ОписаниеТипа = Новый ОписаниеТипов("Число");  
КоличествоДнейРазрешения = ОписаниеТипа.ПривестиЗначение(Значение);

### Ограничение на использование оператора Перейти

#std547

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1. В коде на встроенном языке не рекомендуется использовать оператор **Перейти**, так как необдуманное использование данного оператора приводит к получению запутанных, плохо структурированных модулей, по тексту которых затруднительно понять порядок исполнения и взаимозависимость фрагментов. Вместо оператора **Перейти** рекомендуется использовать другие конструкции встроенного языка.  Например, неправильно:   Если ПланВидовРасчета = Объект.ПланВидовРасчета Тогда      Перейти ~ПланВидовРасчета;     КонецЕсли;  правильно   Если ПланВидовРасчета = Объект.ПланВидовРасчета Тогда      ОбработатьПланВидовРасчета();     КонецЕсли; |

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  2. Запрещается использовать оператор **Перейти** в общих модулях с признаком "Клиент (управляемое приложение)", модулях команд и в клиентском коде модулей управляемых форм, так как данный метод не поддерживается платформой **1С:Предприятие** в режиме веб-клиента. |

## Использование прикладных объектов и универсальных коллекций значений

### Поиск в коллекциях значений

#std452

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При двух и более операциях поиска в объекте **ТаблицаЗначений** с большим количеством строк([[1]](#footnote-1)) рекомендуется:

* Индексировать колонки, по которым выполняется поиск;
* Но только те из них, которые обладают хорошей селективностью (т.е. каждому значению этой колонки должно соответствовать небольшое количество строк). В противном случае, индексирование не даст эффекта, либо он будет отрицательным (потрачено лишнее время на индексирование).

2. Для поиска значений предусмотрены два метода объекта **ТаблицаЗначений**:

* **Найти**
* **НайтиСтроки**

При поиске значения в одной колонке таблицы значений оба метода одинаково эффективно используют индекс, если он был задан (см. п.1).  
Однако при поиске значения сразу по нескольким (или по всем) колонкам необходимо учитывать следующие ограничения.

2.1. Не следует использовать метод **Найти** для поиска по нескольким колонкам в таблицах значений с большим количеством строк, даже если проиндексированы все колонки, обладающие хорошей селективностью. Это ограничение вызвано тем, что метод **Найти** выполняет поиск с применением индекса только по одному полю.  
Например:

ТЗ.Индексы.Добавить("Колонка1");  
ТЗ.Индексы.Добавить("Колонка2");  
... = ТЗ.Найти("найдется все", "Колонка1, Колонка2"); // Индекс НЕ используется!

В этом примере, несмотря на наличие индекса для колонок **Колонка1** и **Колонка2**, поиск все равно будет выполняться перебором всех строк в таблице значений (что очень ресурсоемко на больших объемах данных).

2.2. При использовании метода **НайтиСтроки** в таблицах значений с большим количеством строк следует обеспечить, чтобы список полей индекса был точно таким же, как он задан в структуре поиска (порядок полей не важен). В противном случае, индекс не будет задействован, и поиск будет выполняться перебором всех строк в таблице значений (что очень ресурсоемко на больших объемах данных).  
Например:

ТЗ.Индексы.Добавить("Колонка1"); // Индекс1  
ТЗ.Индексы.Добавить("Колонка2"); // Индекс2  
  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка1, Колонка2 ", "Ищу1","Ищу2")); // Индекс НЕ используется!  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка1", "Ищу1") ); // OK - используется Индекс1  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка2", "Ищу2") ); // OK - используется Индекс2

Другой пример:

ТЗ.Индексы.Добавить("Колонка1,Колонка2");  
  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка1, Колонка2", "Ищу1","Ищу2")); // OK - индекс используется  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка2, Колонка1", "Ищу2","Ищу1")); // OK - индекс используется  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка1", "Ищу1") ); // Индекс НЕ используется!  
... = ТЗ.НайтиСтроки(Новый Структура("Колонка2", "Ищу2") ); // Индекс НЕ используется!

2.3. Аналогичное ограничение действует и для метода **Скопировать** таблицы значений при вызове с параметром **ПараметрыОтбора** (**Структура**).

3. В тех случаях, когда для таблицы значений применяется сортировка по колонкам, содержащим ссылочные значения, необходимо учитывать, что при этом для каждой из этих колонок для всех строк таблицы значений системой будет выполнено обращение к информационной базе за представлением этой ссылки.

Поэтому рекомендуется:

* В тех случаях, когда требуется сортировка по наименованию – сразу, на этапе заполнения, добавлять в таблицу дополнительные колонки с представлениями, и сортировку выполнять уже по ним. Если, конечно, это не вызовет аналогичных многократных обращений к информационной базе;
* В остальных случаях – сортировать «по ссылке», а не по представлению. Для этого в методе **Сортировать** следует использовать объект **СравнениеЗначений**:

ОбъектСравнения = Новый СравнениеЗначений;  
ТаблицаДокументов.Сортировать("Дата,Ссылка", ОбъектСравнения);

Особенно это важно для таблиц с большим количеством (несколько сотен и тысяч) строк, в алгоритмах критических ко времени исполнения.

3.1. При поиске в объекте **Массив** с большим количеством элементов([[2]](#footnote-2)) следует отказаться от массива в пользу:

* объекта **Соответствие**, если не важен порядок элементов;
* индексированной **ТаблицаЗначений**, если порядок элементов значим.

Это обусловлено тем, что в указанных случаях поиск занимает в большинстве случаев константное время, а в массиве поиск выполняется перебором и поэтому пропорционален количеству элементов.

3.2. При необходимости обеспечить уникальность элементов в большом массиве следует однократно в конце алгоритма вызвать функцию **СвернутьМассив** или процедуру **ДополнитьМассив** с параметром**ТолькоУникальныеЗначения = Истина** (модуль **ОбщегоНазначения** Библиотеки стандартных подсистем).

4. Аналогичный недостаток существует и у объекта **ДеревоЗначений**, в котором не предусмотрено индексов и поиск выполняется перебором (как в массиве). В указанных выше случаях объект **ДеревоЗначений** следует заменять индексированным объектом **ТаблицаЗначений**.

### Использование объекта РегистрСведенийМенеджерЗаписи

#std447

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Чтение записи (набора записей) из регистра сведений без последующей модификации необходимо выполнять запросом.  
  
2. Объект **РегистрСведенийМенеджерЗаписи** следует применять только тогда, когда выполнение операций с регистром сведений требует использования отбора одновременно по всем измерениям. При этом менеджер записи использует для выполнения записи два набора записей, устанавливая им соответствующие значения отборов. Поэтому обработчики событий набора записей вызываются и тогда, когда для записи данных используется менеджер записи.  
  
3. В остальных случаях следует использовать объект **РегистрСведенийНаборЗаписей**. С точки зрения производительности использование менеджера записей в некоторых случаях будет столь же эффективным, как и использование набора записей, а в некоторых - менее, так как будут выполняться лишние действия.  
  
Правильно:

Набор = РегистрыСведений.ЗначенияПравПользователя.СоздатьНаборЗаписей();  
Набор.Отбор.НаборПрав.Установить(ЗначениеНабораПрав);  
Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав Цикл  
  Запись = Набор.Добавить();  
  Запись.НаборПрав = ЗначениеНабораПрав;  
  Запись.Право = СтрокаТаблицы.Право;  
  Запись.Значение = СтрокаТаблицы.Значение;   
КонецЦикла;  
Набор.Записать();

Неправильно:

Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав Цикл  
  ЭлементРегистраСведений = РегистрыСведений.ЗначенияПравПользователя.СоздатьМенеджерЗаписи();  
  ЭлементРегистраСведений.НаборПрав = ЗначениеНабораПрав;  
  ЭлементРегистраСведений.Право = СтрокаТаблицы.Право;  
  ЭлементРегистраСведений.Значение = СтрокаТаблицы.Значение;  
  ЭлементРегистраСведений.Записать();  
КонецЦикла;

### Копирование строк между таблицами значений (табличными частями и т.п.) произвольной структуры

#std448

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

При копировании строк между различными таблицами значений (табличными частями и т.п.) со схожим составом колонок следует использовать метод глобального контекста **ЗаполнитьЗначенияСвойств.**  
  
Алгоритмы, использующие данный метод значительно эффективнее, чем например, многократный перебор колонок таблицы значений, выполняемый для получения их состава.  
  
Правильно:

Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл  
  СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();  
  ЗаполнитьЗначенияСвойств(СтрокаТаблицыПриемника, СтрокаТаблицыИсточника);  
КонецЦикла;

Неправильно:

Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл  
  СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();  
  Для каждого Колонка Из ТаблицаПриемник.Колонки Цикл  
    КолонкаТаблицыИсточника = ТаблицаИсточник.Колонки.Найти(Колонка.Имя);  
    Если КолонкаТаблицыИсточника <> Неопределено Тогда  
      СтрокаТаблицыПриемника[Колонка.Имя] = СтрокаТаблицыИсточника[Колонка.Имя];  
    КонецЕсли;  
  КонецЦикла;  
КонецЦикла;

### Порядок записи движений документов

#std450

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Не рекомендуется использовать явную запись наборов записей регистров (с помощью метода **Записать**) в процедурах обработки проведения документов. Запись должна производится неявно системой, при завершении процедуры проведения.  
  
В случае же нарушения этого правила, при параллельной работе нескольких пользователей, возможна ситуация возникновения взаимных блокировок при проведении документов.  
  
2. Исключением является ситуация, когда данные, сохраняемые в регистрах, необходимы в последующих алгоритмах, выполняемых до момента выхода из процедуры проведения.

См. также

* [Требования к проведению документов](https://its.1c.ua/db/content/v8std/src/100/300/i8100603.htm?_=1580137113)
* [Самодостаточность регистров](https://its.1c.ua/db/content/v8std/src/100/300/i8100477.htm?_=1580137113)
* [Использование активности движений](https://its.1c.ua/db/content/v8std/src/100/300/i8100633.htm?_=1580137113)

### Получение представлений для ссылочных значений в табличном документе

#std449

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  При формировании табличного документа запрещено в качестве параметров ячеек с типом заполнения **Параметр** указывать ссылочные значения, поскольку в этом случае в момент вывода данных в табличный документ будет выполнено многократное обращение к базе данных для получения представлений этих значений. Поэтому в качестве параметров следует указывать сами представления.  Исключением могут быть случаи, когда для получения представлений придется выполнять аналогичное многократное обращение к базе данных.  При этом следует иметь ввиду, что при получении представлений для полей непосредственно в самом запросе (через поле **Представление** или функцией **Представление**(<Имя поля>)) выполняется неявное соединение с таблицей объекта, для которого получаются представления. Для полей составного типа - несколько соединений, для каждого из типов, входящих в состав. Это может приводить к увеличению времени выполнения запроса (и как следствие, общего времени формирования итогового документа), а при большом количестве типов – к невозможности его выполнения в клиент-серверной версии из-за ограничения **Microsoft SQL Server 2005**, по которому в запросе не может участвовать больше 256 таблиц. Такие случаи также могут быть исключением для данного правила, в них представления для ссылочных значений допускается получать в момент их вывода в табличный документ.  Поскольку однозначно рекомендовать - какой из способов получения представлений следует выбрать - нельзя, такой выбор должен делаться разработчиком самостоятельно, на основании данных, полученных экспериментально.  См. также   * [Особенности работы с полем Представление и функцией Представление() языка запросов (статья на ИТС)](http://its.1c.ru/db/metod81#content:2655:1) * [Вывод ссылочных полей](http://its.1c.ru/db/metod81#content:2656:1) |

### Программное создание прикладных объектов

#std451

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Для программного создания прикладных объектов следует использовать методы соответствующих менеджеров (**СоздатьЭлемент**, **СоздатьДокумент**, **СоздатьНаборЗаписей** и т.д.).

Для программного создания прикладных объектов, у которых существует соответствующие менеджеры объектов, использование конструктора (оператор встроенного языка **Новый**) запрещается.  
  
Правильно:

ДокументПриходная = Документы.ПоступлениеТоваровУслуг.СоздатьДокумент();

Неправильно:

ДокументПриходная = Новый("ДокументОбъект.ПоступлениеТоваровУслуг");

2. При программном создании объекта следует явно вызывать метод объекта **Заполнить**. Если данных для заполнения нет, то передать значение **Неопределено**. В этом случае корректно отработают свойства реквизитов объекта **"Значение заполнения"**, будет вызван обработчик **ОбработкаЗаполнения** и подписки на это событие, как при интерактивной работе с объектом.

Например, неправильно:

Папка = Справочники.ПапкиФайлов.СоздатьЭлемент();  
// ...  
Папка.Записать();

Правильно:

Папка = Справочники.ПапкиФайлов.СоздатьЭлемент();  
// ...  
Папка.Заполнить(Неопределено);  
// ...  
Папка.Записать();

Исключением могут быть случаи, когда объект полностью загружается из источника при обмене данными или восстановление базы из резервной копии (загрузка из XML).

### Использование модуля объекта, модуля менеджера объекта и общих модулей

#std486

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  1. **Модуль объекта** предназначен для реализации поведения отдельного экземпляра объекта (**СправочникОбъект**, **ДокументОбъект** и т.п.). В модуле объекта размещаются процедуры и функции, которые работают с данными объекта (**ЭтотОбъект**и переменные модуля объекта), в том числе когда он еще не записан в информационную базу.  Например, в модуле объекта могут размещаться:   * обработчики событий объекта * процедуры заполнения экземпляра объекта.   Следует иметь в виду, что для вызова экспортных процедур и функций модуля объекта из других модулей может потребоваться предварительно получить сам экземпляр объекта из информационной базы с помощью метода **ПолучитьОбъект**. При этом происходит загрузка объекта из базы целиком, вместе с его табличными частями, что достаточно ресурсоемко.  См. также: [Чтение отдельных реквизитов объекта из базы данных](https://its.1c.ua/db/content/v8std/src/200/300/i8100496.htm?_=1580137113)  2. **Модуль менеджера** объекта предназначен для размещения "статической" функциональности, которая логически неразрывно связана с объектом метаданных, но не зависит от состояния конкретного экземпляра объекта данных. Это могут быть процедуры и функции:   * относящиеся не к одному, а сразу к некоторой совокупности объектов. Например, это функции для вывода на печать списка объектов; функции, возвращающие информацию, общую для всех экземпляров объекта метаданных; процедуры обновления данных информационной базы, которые связаны с объектом метаданных; и т.п. * которые работают с объектом, записанным в ИБ. В таких функциях входным параметром является ссылка на объект. Например, это функции для получения печатной формы по ссылке на объект, процедуры формирования движений по ссылке на объект и т.п.   Для выполнения функций модуля менеджера объекта не должен требоваться экземпляр объекта данных (**СправочникОбъект**, **ДокументОбъект** и т.п.).  3. Если функциональность невозможно однозначно отнести к тому или иному объекту метаданных, то она является логически общей для нескольких объектов. В этом случае ее следует размещать в **общем модуле**. |

### Ограничения на использование экспортных процедур и функций

#std544

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Не следует размещать экспортные процедуры и функции в модулях команд и общих команд. К этим модулям нет возможности обращаться из внешнего по отношению к ним кода, поэтому экспортные процедуры и функции в этих модулях не имеют смысла.

### Установка параметров выбора и связей параметров выбора для объектов метаданных

#std411

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Как правило, такие ограничения бизнес-логики как ограничения выбора, должны быть одинаковыми для всех форм, в которых редактируется тот или иной объект. Поэтому задавать параметры выбора и связи параметров выбора рекомендуется в свойствах объектов метаданных - в реквизитах справочников, документов и т.п.  
  
2. Однако могут встречаться случаи, когда ограничения выбора могут зависеть от конкретного сценария работы. В таких случаях параметры выбора могут быть уточнены по месту, в конкретной форме.  
  
Например, в конфигурации имеются

* справочник **Сотрудники**, в котором есть реквизиты **Организация** и **ВидСотрудника** (перечисление со значениями **Основной**/**Совместитель**);
* документ **ПриказОПриеме**, в котором есть реквизит **Организация** и **Сотрудник**; при этом для реквизита **Сотрудник** документа **ПриказОПриеме** задана связь параметра выбора **Отбор.Организация** с реквизитом **Организация**.

Требуется реализовать возможность выбора только основных сотрудников в зависимости от значения функциональной опции **ВыбратьТолькоИзОсновыхСотрудников**. Для этого необходимо

* в форме документа **ПриказОПриеме** реализовать дополнительный реквизит формы **ВидыСотрудников**,
* после чего для поля формы **Сотрудник**может быть установлена связь параметра выбора **Отбор.ВидСотрудника** с реквизитом формы **ВидыСотрудников**,
* при этом реквизит формы **ВидыСотрудников**заполняется на основании анализа функциональной опции.

(При этом установить для реквизита **Сотрудник** документа **ПриказОПриеме** связь для параметра выбора **Отбор.ВидСотрудника** нет возможности, т.к. реквизита **ВидСотрудника** в документе **ПриказОПриеме** не существует.)  
  
Тогда установка связи для параметра **Отбор.Организация** в свойствах реквизита **Сотрудник** документа **ПриказОПриеме** и связи для параметра **Отбор.ВидСотрудника** в поле формы документа **ПриказОПриеме** приведет к тому, что в режиме **1С:Предприятия** будут работать обе связи параметра выбора. Таким образом, при выборе сотрудника в форме приказа о приеме отбор в списке сотрудников будет установлен как по организации, заполненной в документе, так и по виду сотрудника, который будет определен на основании функциональной опции.

### Использование РеквизитФормыВЗначение и ДанныеФормыВЗначение

#std409

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

|  |
| --- |
| Методическая рекомендация (полезный совет)  В большинстве случаев, в модулях форм следует использовать метод формы **РеквизитФормыВЗначение** вместо метода **ДанныеФормыВЗначение**.  Рекомендация обусловлена соображениями унификации прикладного кода и тем, что синтаксис метода **РеквизитФормыВЗначение** проще, чем у **ДанныеФормыВЗначение**(а следовательно, меньше вероятность ошибки). В **ДанныеФормыВЗначение**необходимо дополнительно передавать тип значения:  ТаблицаПодписей = ДанныеФормыВЗначение(ТаблицаПодписей, Тип("ТаблицаЗначений"));  а для **РеквизитФормыВЗначение** это не обязательно, а в практическом плане - избыточно:  ТаблицаПодписей = РеквизитФормыВЗначение("ТаблицаПодписей");  Наличие в платформе **1С:Предприятие** метода формы **РеквизитФормыВЗначение** (наряду с методом глобального контекста **ДанныеФормыВЗначение**) объясняется только удобством его применения. С точки зрения эффективности и результата методы работают одинаково. |

### Применение параметров отчета в СКД

#std407

Область применения: управляемое приложение, обычное приложение.

При использовании параметров в отчетах, реализуемых с помощью системы компоновки данных, следует придерживаться следующих рекомендаций.

* Избегать использования параметров отчета в том случае, если можно обойтись "обычным" управлением элементами отбора. Например, не применять в запросах СКД условия типа:
* ГДЕ Организация = &Организация
* Взамен этого дать пользователю возможность управлять элементом отбора **Организация**.
* Избегать применения обязательных параметров для того, чтобы исключить ситуацию, когда пользователь может отключить параметры отчета, ожидая отмены их применения, и получить при этом ошибку выполнения "Не задан параметр отчета". Для этого следует  
  использовать синтаксические элементы расширения языка запросов системы компоновки данных **{ГДЕ...}**.  
    
  Например, применение конструкции

{ГДЕ Сведения.Период >= &ДатаНачала, Сведения.Период <= &ДатаОкончания}

приведет к тому, что оба параметра отчета **ДатаНачала** и **ДатаОкончания** будут необязательными и при их отключении пользователем будут отключены соответствующие фрагменты условия.

### Использование объектов типа Структура

#std693

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Требования, предъявляемые данным стандартом к **Структурам**, направлены на повышение читаемости кода и упрощение внесения изменений в код разными авторами (разработчиками) как при коллективной разработке, так и при доработке прикладных решений на конкретных внедрениях. Повышение читаемости кода в свою очередь ведет к уменьшению допускаемых при разработке ошибок и повышает качество прикладного решения.

1. При создании объекта типа **Структура** не рекомендуется передавать в конструктор более 3-х значений свойств. Вместо этого рекомендуется использовать метод **Вставить** или присваивать значения свойствам явным образом

Неправильно

ПараметрыФормыКомпоновки  = Новый Структура(  
   "НеПомещатьНастройкиВСхемуКомпоновкиДанных,  
   |НеРедактироватьСхемуКомпоновкиДанных,  
   |НеНастраиватьУсловноеОформление,  
   |НеНастраиватьВыбор,  
   |НеНастраиватьПорядок,  
   |АдресСхемыКомпоновкиДанных,  
   |АдресНастроекКомпоновкиДанных,  
   |УникальныйИдентификатор,  
   |Заголовок",  
   Истина,  
   Истина,  
   Истина,  
   Истина,  
   Истина,  
   ТекущиеДанные.АдресСхемыКомпоновкиДанных,  
   ?(АдресНастроекСхемыКомпоновкиДанных <> Неопределено,  
        АдресНастроекСхемыКомпоновкиДанных,  
        ТекущиеДанные.АдресНастроекСхемыКомпоновкиДанных),  
   УникальныйИдентификатор,  
   ЗаголовокФормыНастройкиСхемыКомпоновкиДанных));

Правильно

ПараметрыФормыКомпоновки  = Новый Структура;

ПараметрыФормыКомпоновки.Вставить("НеПомещатьНастройкиВСхемуКомпоновкиДанных", Истина);  
ПараметрыФормыКомпоновки.Вставить("НеРедактироватьСхемуКомпоновкиДанных", Истина);  
ПараметрыФормыКомпоновки.Вставить("НеРедактироватьСхемуКомпоновкиДанных", Истина);  
ПараметрыФормыКомпоновки.Вставить("НеНастраиватьВыбор", Истина);  
ПараметрыФормыКомпоновки.Вставить("НеНастраиватьПорядок", Истина);  
ПараметрыФормыКомпоновки.Вставить("АдресСхемыКомпоновкиДанных", ТекущиеДанные.АдресСхемыКомпоновкиДанных);  
ПараметрыФормыКомпоновки.Вставить("АдресНастроекКомпоновкиДанных", ?(АдресНастроекСхемыКомпоновкиДанных <> Неопределено,  
                                                                                                                         АдресНастроекСхемыКомпоновкиДанных,  
                                                                                                                         ТекущиеДанные.АдресНастроекСхемыКомпоновкиДанных));  
ПараметрыФормыКомпоновки.Вставить("УникальныйИдентификатор ", УникальныйИдентификатор);  
ПараметрыФормыКомпоновки.Вставить("Заголовок", ЗаголовокФормыНастройкиСхемыКомпоновкиДанных);

2. Не рекомендуется в конструкторе структуры использовать конструкторы других объектов, если эти конструкторы принимают параметры. В частности в конструкторе одной структуры не рекомендуется создавать другие структуры с объявлением значений свойств.

Неправильно

НоменклатураСервер.ЗаполнитьСлужебныеРеквизитыПоНоменклатуреВКоллекции(  
  Объект.Товары,  
  Новый Структура(  
  "ЗаполнитьПризнакХарактеристикиИспользуются,  
  |ЗаполнитьПризнакТипНоменклатуры, ЗаполнитьПризнакВариантОформленияПродажи",  
   Новый Структура("Номенклатура", "ХарактеристикиИспользуются"),  
   Новый Структура("Номенклатура", "ТипНоменклатуры"),  
   Новый Структура("Номенклатура", "ВариантОформленияПродажи")  
  )  
 );

Правильно

ПараметрыЗаполненияРеквизитов = Новый Структура;  
ПараметрыЗаполненияРеквизитов.Вставить("ЗаполнитьПризнакХарактеристикиИспользуются",  
                                                          Новый Структура("Номенклатура", "ХарактеристикиИспользуются"));  
ПараметрыЗаполненияРеквизитов.Вставить("ЗаполнитьПризнакТипНоменклатуры",  
                                                          Новый Структура("Номенклатура", "ТипНоменклатуры"));  
НоменклатураСервер.ЗаполнитьСлужебныеРеквизитыПоНоменклатуреВКоллекции(Объект.Товары,   
                                                          ПараметрыЗаполненияРеквизитов);

3. Не рекомендуется в конструкторе структуры вызывать функции с большим (более 3) количеством параметров.

Неправильно

СведенияОТоваре = Новый Структура("ПараметрыТовара, ЦенаПродажиИОстаткиТовара, ЦенаЗакупкиИОстаткиТовара",  
     ПодборТоваровКлиентСервер.ПараметрыТовара(),  
     ПодборТоваровВызовСервера.ЦенаПродажиИОстаткиТовара(  
      Номенклатура,  
      Характеристика,  
      Соглашение,  
      Валюта,  
      ВидыЦен),  
     ЦенаЗакупкиИОстаткиТовара(  
      Номенклатура,  
      Характеристика,  
      Соглашение,  
      Валюта,  
      ВидыЦен));

Правильно

СведенияОТоваре = Новый Структура("ПараметрыТовара, ЦенаПродажиИОстаткиТовара, ЦенаЗакупкиИОстаткиТовара");  
СведенияОТоваре.ПараметрыТовара           = ПодборТоваровКлиентСервер.ПараметрыТовара();  
СведенияОТоваре.ЦенаПродажиИОстаткиТовара = ПодборТоваровВызовСервера.ЦенаПродажиИОстаткиТовара(  
        Номенклатура,  
        Характеристика,  
        Соглашение,  
        Валюта,  
        ВидыЦен);  
СведенияОТоваре.ЦенаПродажиИОстаткиТовара = ЦенаЗакупкиИОстаткиТовара.ЦенаЗакупкиИОстаткиТовара(  
        Номенклатура,  
        Характеристика,  
        Соглашение,  
        Валюта,  
        ВидыЦен);

См. также

* [Параметры процедур и функций](https://its.1c.ua/db/content/v8std/src/300/100/i8100640.htm?_=1580137113)
* [Особенности использования структур в качестве параметров процедур и функций](https://its.1c.ua/db/content/v8std/src/300/100/i8100641.htm?_=1580137113)

### Особенности сортировки в таблице значений

#std781

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

В тех случаях, когда для таблицы значений применяется сортировка по колонкам, содержащим ссылочные значения, необходимо учитывать, что при этом для каждой из этих колонок для всех строк таблицы значений системой будет выполнено обращение к информационной базе за представлением этой ссылки.

Поэтому рекомендуется:

* В тех случаях, когда требуется сортировка по наименованию – сразу, на этапе заполнения, добавлять в таблицу дополнительные колонки с представлениями, и сортировку выполнять уже по ним. Если, конечно, это не вызовет аналогичных многократных обращений к информационной базе;
* В остальных случаях – сортировать «по ссылке», а не по представлению. Для этого в методе **Сортировать** следует использовать объект **СравнениеЗначений**:

ОбъектСравнения = Новый СравнениеЗначений;  
ТаблицаДокументов.Сортировать("Дата,Ссылка", ОбъектСравнения);

Особенно это важно для таблиц с большим количеством (несколько сотен и тысяч) строк, в алгоритмах критических ко времени исполнения.

См. также

* [Упорядочивание результатов запроса](https://its.1c.ua/db/content/v8std/src/200/100/i8100412.htm?_=1580137113)

### Массовая конкатенация строк

#std782

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

При массовых([[3]](#footnote-3)) операциях конкатенации строк следует использовать методы платформы **СтрРазделить** и **СтрСоединить**.  
Например, неправильно:

Для НомерКолонки = 1 По Макет.ШиринаТаблицы Цикл  
  ИзвлеченныйТекст = ИзвлеченныйТекст + Символы.ПС + ТекстОбласти;  
  …

Правильно:

ИзвлеченныеТексты = Новый Массив;  
Для НомерКолонки = 1 По Макет.ШиринаТаблицы Цикл  
  ИзвлеченныеТексты.Добавить(ТекстОбласти);  
  …  
ИзвлеченныйТекст = СтрСоединить(ИзвлеченныеТексты, Символы.ПС);

Такая обработка данных не только быстрее выполняется, но и приводит к снижению потребления оперативной памяти.

См. также

* [Оформление текстов запросов](https://its.1c.ua/db/content/v8std/src/200/100/i8100437.htm?_=1580137113) (раздел "Конкатенация нескольких текстов запросов в пакет")

# [Клиент-серверное взаимодействие](https://its.1c.ua/db/v8std/browse/13/-1/35)

## Использование модулей с повторным использованием возвращаемых значений

#std724

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Общие модули с повторным использованием возвращаемых значений (далее: кэш) предусмотрены для кэширования результатов работы функций, которые в них размещены - на время сеанса или на время вызова. Их следует применять для экономии вычислительных ресурсов сервера и для [минимизации клиент-серверного взаимодействия](https://its.1c.ua/db/content/v8std/src/400/i8100487.htm?_=1580137113).

См. также: раздел "[*Повторное использование возвращаемых значений*](http://its.1c.ru/db/v83doc/bookmark/dev/ti000000181)" документации по платформе 1С:Предприятие,[*Использование значений, влияющих на поведение клиентского приложения*](https://its.1c.ua/bmk/std/common_values)

2. В то же время, чрезмерное (неоправданное) применение общих модулей с повторным использованием возвращаемых значений может приводить к [излишнему потреблению памяти](https://its.1c.ua/db/content/v8std/src/400/i8100725.htm?_=1580137113).

2.1. Недопустимо создать общие модули с повторным использованием, из которых возвращаются данные, вычисление которых выполняется быстрее, чем получение из кэша. Например, строковые константы. Кроме того, что получение строковой константы каждый раз будет работать гораздо быстрее, чем получение ее из общего модуля с повторным использованием, эти данные будут занимать память кэша.  
Например, неправильно размещать в модуле с повторным использованием:

Функция ИмяПакетаУправления() Экспорт  
  Возврат "ManagementPackage";  
КонецФункции

Имеет смысл кэшировать данные, полученные из базы данных, внешних источников данных или путем сложных (ресурсоемких) вычислений. Причем в ряде случаев, даже значения, полученные из базы данных, не стоит кэшировать, если выгода от их кэширования – неочевидна. Например, не стоит кэшировать [константы (объект метаданных)](https://its.1c.ua/db/content/v8std/src/100/300/i8100632.htm?_=1580137113) примитивных типов, поскольку часто они привносят лишь незначительную долю от общего времени выполнения ресурсоемкой операции.

2.2. Следует помещать в кэш только такие данные, к которым потом будут часто обращаться.

В частности, следует иметь в виду, что кэш не хранит данные вечно. Закэшированное значение будет удалено из кэша через 20 минут после вычисления или через 6 минут после последнего использования (в зависимости от того, что наступит раньше\*). Кроме этого значение будет удалено при нехватке оперативной памяти в рабочем процессе сервера, при перезапуске рабочего процесса и при переключении клиента на другой рабочий процесс. Поэтому если никто "не успел" воспользоваться данными из кэша, то этот ресурс был потрачен зря.

\* Примечание:  конкретные цифры могут варьироваться в зависимости от используемой версии платформы 1С:Предприятие.

2.3. Диапазон значений входных параметров функций, размещенных в общих модулей с повторным использованием, не должен быть широким.  
Например, в конфигурации предусмотрена функция, получающая на вход контрагента. Если контрагентов в базе очень много, а сценарий работы пользователей таков, что вероятность того, что кто-то за 5 минут обратится к этому же контрагенту, очень невысокая, то ресурсы будут потрачены впустую. Кроме того, если эту «трату» умножить на количество одновременно работающих пользователей, то бесполезные расходы ресурсов становятся значительными.

3. Не следует изменять данные, полученные  из кэша. В противном случае, возможны скрытые ошибки в работе программы, а также бесполезное расходование памяти (ресурсов кэша). Поэтому в качестве возвращаемых значений рекомендуется  использовать значения, состояние которых изменить нельзя, например: **ФиксированныйМассив**, **ФиксированнаяСтруктура**.

Это ограничение вызвано тем, что кэш возвращает каждый раз не копию объекта, а ссылку на один и тот же объект в памяти. Например, если в массив, который возвращает функция с повторным использованием, при каждом вызове при проведении документов дописывать новое значение, то в результате кэш очень быстро «распухнет». Кроме того, при очередном сбросе кэша, добавленные значения будут потеряны и код, который на них опирался, будет работать некорректно.

4. Если в модуле с повторным использованием размещено несколько экспортных функций, которые не только вызываются «снаружи», но и вызывают друг друга, то следует иметь в виду, что результат «внутренних» вызовов не кэшируется.  
Например, если в модуле **ОбменДаннымиПовтИсп** размещено две экспортных функции:

Функция АвтономнаяРаботаПоддерживается() Экспорт  
 Возврат ...  
КонецФункции

Функция ЭтоАвтономноеРабочееМесто() Экспорт  
 Возврат АвтономнаяРаботаПоддерживается() И ...;  
КонецФункции

которые последовательно вызываются из прикладного кода,

... = ОбменДаннымиПовтИсп.АвтономнаяРаботаПоддерживается();  
... = ОбменДаннымиПовтИсп.ЭтоАвтономноеРабочееМесто();

то функция **АвтономнаяРаботаПоддерживается** будет вычислена дважды.

Для того чтобы ее возвращаемое значение всегда получалось из кэша, следует явно указывать имя модуля:

Функция ЭтоАвтономноеРабочееМесто() Экспорт  
 Возврат ОбменДаннымиПовтИсп.АвтономнаяРаботаПоддерживается() И ...;  
КонецФункции

5. Если у общего модуля свойство "Повторное использование возвращаемых значений" установлено в значение "На время сеанса", то в значениях, возвращаемых функциями такого модуля, нельзя использовать значения типа **МенеджерВременныхТаблиц**, **Запрос**, объекты базы данных (например, **ДокументОбъект**, **ОтчетОбъект**) причем, как непосредственно, так и в составе любых коллекций. Ограничение вызвано тем, что значения этих типов допустимо использовать только в том же серверном вызове, в котором они были получены (созданы).

Возврат значений этих типов в указанных функциях не проверяется платформой и приводит к трудно диагностируемой остановке работы программы (записи есть только в технологическом журнале).

Данное ограничение распространяется на использование значений во временном хранилище.

## Использование значений, влияющих на поведение клиентского приложения

#std459

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

В том случае, если поведение целого ряда форм или команд зависит от одних и тех же значений (параметров, задаваемых пользователем или иных настроек поведения клиентского приложения), для [минимизации клиент-серверного взаимодействия](https://its.1c.ua/db/content/v8std/src/400/i8100487.htm?_=1580137113) рекомендуется применять [общие модули с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100724.htm?_=1580137113) (свойство «Повторное использование возвращаемых значений» равно «На время сеанса»). Применение таких общих модулей допускается, если изменение соответствующего значения в течение сеанса не является критичным, т.е. в течение всего сеанса может использоваться однажды полученное значение.  
  
При этом функции такого общего модуля должны за один вызов возвращать сразу все значения, которые могут понадобиться в тех или иных обстоятельствах.  
  
Например, если при работе всех форм подсистемы регистрации занятости сотрудников, требуются пользовательские параметры «Время занятости по умолчанию», «Время начала рабочего дня» и «Время окончания рабочего дня», то все эти параметры необходимо получать одним вызовом, возвращающим структуру с тремя свойствами:

Настройки = НастройкиПодсистемыРегистрации();  
ВремяНачала = Настройки.ВремяНачалаРабочегоДня;  
ВремяОкончания = Настройки.ВремяОкончанияРабочегоДня;

Следует помнить, что применение клиентских общих модулей с повторным использованием возвращаемых значений должно быть разумно ограничено. В частности, данные, влияющие на поведение отдельных форм, рекомендуется размещать в реквизитах самой формы, получая их при создании формы на сервере. И только если от некоторого значения (или нескольких значений) зависит поведение команд или большого числа форм приложения, следует применять общие модули с повторным использованием возвращаемых значений.  
  
Не рекомендуется использовать переменные модуля управляемого приложения и модуля обычного приложения для минимизации клиент-серверного взаимодействия.

## Получение предопределенных значений на клиенте

#std443

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

Для получения значения ссылок на предопределенные элементы справочников, планов видов характеристик, планов счетов, планов видов расчета, ссылки на значения перечислений, системных перечислений и точек маршрута бизнес-процессов в коде, выполняемом «на стороне» тонкого клиента, веб-клиента, где объекты типа **СправочникМенеджер.<ИмяСправочника>**, **ПеречислениеМенеджер.<ИмяПеречисления>** и т.п. не доступны, предназначена функция глобального контекста **ПредопределенноеЗначение**.  
Например:

ЮрФизЛицо = ПредопределенноеЗначение("Перечисление.ЮридическоеФизическоеЛицо.ЮридическоеЛицо");

При этом не следует в прикладном коде реализовывать собственные механизмы кеширования на клиенте предопределенных значений. Функция **ПредопределенноеЗначение** не ухудшает [клиент-серверное взаимодействие](https://its.1c.ua/db/content/v8std/src/400/i8100487.htm?_=1580137113): серверный вызов выполняется только при первом обращении к значению, а результат автоматически кешируется в кеше конфигурации на клиенте до следующего обновления версии конфигурации или версии платформы.

Подробнее см. раздел [Работа с предопределенными значениями](http://its.1c.ru/db/v83doc#bookmark:dev:TI000000158) в документации к платформе 1С:Предприятие (на ИТС).

См. также

* [Использование предопределенных элементов](https://its.1c.ua/db/content/v8std/src/100/300/i8100697.htm?_=1580137113)

## Минимизация количества серверных вызовов

#std487

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Разработку управляемого приложения необходимо вести с контролем количества вызовов серверных процедур и функций из клиентского кода (серверных вызовов), а в некоторых случаях – также объем передаваемых данных между клиентом и сервером (трафик).

Общее количество серверных вызовов складывается из

* обращений на сервер, которые выполняет платформа **1С:Предприятие**,
* и вызовов, которые выполняются из клиентского кода конфигурации – отклонение, привносимое конфигурацией по отношению к платформе.

В общем виде, при проектировании клиент-серверного взаимодействия в конфигурации следует руководствоваться тем, что на каждое действие пользователя в клиентском коде конфигурации не должно выполняться дополнительных вызовов на сервер. Любые исключения из этого правила требуют дополнительного обоснования.

1.2. Отдельно для режима с низкой скоростью соединения следует оптимизировать не только количество вызовов, но и объем передаваемых данных между клиентом и сервером (трафик). Отладку клиент-серверного взаимодействия в этом режиме работы рекомендуется проводить в режиме имитации задержек серверных вызовов.

Ниже рассмотрены типовые действия пользователя и даны рекомендации по организации клиент-серверного взаимодействия.

### Запуск клиентского приложения

2.1. В простейшем случае, код конфигурации, выполняемый при запуске клиентского приложения, не должен приводить к обращениям на сервер. В тех случаях, когда все же необходимо получать данные с сервера:

* не следует напрямую вызывать серверные процедуры и функции из кода модуля приложения, модуля управляемого приложения и модуля внешнего соединения.
* правильно: за один вызов на сервер передавать на клиент все параметры, необходимые для запуска приложения.

В случае если параметры для запуска клиентского приложения требуется запрашивать с сервера из разных мест клиентского кода, следует разместить такую функцию в общем серверном [модуле с повторным использованием возвращаемых значений](https://its.1c.ua/db/content/v8std/src/400/i8100724.htm?_=1580137113). При первом вызове этой функции происходит одно обращение к серверу, после чего полученное значение автоматически кешируется платформой на клиенте для всех повторных вызовов этой функции.

Пример:

// Фрагмент общего серверного модуля СтандартныеПодсистемыПереопределяемый с повторным использованием возвращаемых значений  
Функция ПараметрыРаботыКлиента()  
  Параметры = Новый Структура();  
  Параметры.Вставить("ИнформационнаяБазаФайловая", ОбщегоНазначения.ИнформационнаяБазаФайловая());  
  // Инициализация других параметров, необходимых на клиенте при запуске приложения  
  // Параметры.Вставить(имя параметра, значение параметра);  
  Возврат Параметры;  
КонецФункции

Пример клиентского кода, использующего функцию **ПараметрыРаботыКлиента**:

ИнформационнаяБазаФайловая = СтандартныеПодсистемыПереопределяемый. ПараметрыРаботыКлиента().ИнформационнаяБазаФайловая;  
Если ИнформационнаяБазаФайловая Тогда  
 // обработка этого случая в клиентском коде  
 //…

См. также: [*Использование значений, влияющих на поведение клиентского приложения*](https://its.1c.ua/db/content/v8std/src/400/i8100459.htm?_=1580137113)

### Открытие управляемой формы

3.1. В случае если открытие формы выполняется из кода, следует открывать форму за один вызов с помощью метода глобального контекста **ОткрытьФорму** (при использовании версии платформы 1С:Предприятие 8.2 и более ранних версий - также **ОткрытьФормуМодально**). Для передачи параметров в форму следует использовать параметр этих методов **Параметры**.

3.2. При открытии формы не допускается выполнять обращений к серверу из кода модуля формы в обработчиках клиентских событий формы, таких как **ПриОткрытии** и **ПриПовторномОткрытии**. При необходимости обращения из них к серверным данным, следует размещать эти данные в реквизитах формы в **ПриСозданииНаСервере**.

Например, неправильно:

НастройкаПроксиСервера = СерверныйМодуль.НастройкаПроксиСервера();  
ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера", Новый Структура("Настройка", НастройкаПроксиСервера));

правильно:

ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера");

при этом получение значения константы выполнять в обработчике **ПриСозданииНаСервере** формы **ПараметрыПроксиСервера**.

### Выполнение локальной команды управляемой формы

4.1. Выполнение локальной команды формы должно приводить не более чем к одному вызову сервера.

* Если команда выполняет только клиентские операции (приводит к открытию новой формы, устанавливает отбор в списке, меняет стиль оформление и пр.), то все необходимые данные для ее выполнение должны быть заранее переданы на клиент. Рекомендуется заранее готовить эти данные в обработчике события формы **ПриСозданииНаСервере** и размещать их в реквизитах формы.  
    
  Пример:  
  При выборе товаров из списка номенклатуры требуется запретить для пользователя выбор групп номенклатуры: при попытке выбрать группу программа должна выдавать останавливающее сообщение. Для проверки, является ли выбранный элемент номенклатуры группой, неправильно вызывать отдельную серверную функцию. Следует добавить в таблицу значений, которая связана с полем списка на форме, реквизит **ЭтоГруппа** и заполнять его в обработчике события формы **ПриСозданииНаСервере** . Тогда проверка на клиенте выполняется без дополнительного серверного вызова и имеет вид:

ТекущаяСтрока = Элемент.ТекущиеДанные;   
Если ТекущаяСтрока.ЭтоГруппа Тогда  
  Сообщение = Новый СообщениеПользователю();  
  Сообщение.Текст = НСтр("ru = 'Выбор группы запрещен.'");  
  Сообщение.Сообщить();  
  Возврат;  
КонецЕсли;

* Если команда связана с выполнением бизнес-логики, которую возможно отработать только на сервере, то вся она должна выполняться за один серверный вызов.

### Выбор из справочника

4.2. В общем случае, при выборе из справочника допускается выполнять только один серверный вызов из кода, к которому приводит вызов метода глобального контекста **ОткрытьФорму** (или **ОткрытьФормуМодально**).

4.3. В случае если после выбора из справочника необходимо выполнить бизнес-логику, которую возможно отработать только на сервере, то допустимо выполнять ее за один дополнительный серверный вызов.

### Выполнение глобальной команды

5.1. При выполнении глобальной команды допускается выполнять только один серверный вызов из кода. В случае если команда открывает форму, то этот вызов должен выполняться при вызове метода глобального контекста **ОткрытьФорму** (или **ОткрытьФормуМодально**).

### Выполнение команды формирования отчета

6.1. При выполнении команды формирования отчета не допускается выполнять дополнительных серверных вызовов из кода конфигурации.

В частности, при открытии формы отчета не допускается выполнять обращений к серверу из кода модуля формы в обработчиках клиентских событий формы, таких как **ПриОткрытии** и **ПриПовторномОткрытии**.

Например, неправильно выполнять формирование отчета, использующего систему компоновки данных, из обработчика формы **ПриОткрытии**:

&НаКлиенте  
Процедура ПриОткрытии(Отказ)  
  ВывестиОтчет();  
КонецПроцедуры

&НаСервере  
Процедура ВывестиОтчет()  
  // код по формированию отчета…  
КонецПроцедуры

правильно:

* открывать форму отчета с параметром **СформироватьПриОткрытии** = **Истина**, либо
* установить параметр формы **СформироватьПриОткрытии** в значение **Истина** в обработчике **ПриСозданииНаСервере**.

### Выполнение подбора элементов

7.1. Особенность клиент-серверного взаимодействия при выполнении подбора элементов состоит в необходимости передавать список выбранных элементов между формой объекта и формой подбора. При этом объем передаваемых данных может быть достаточно большим.

В этом случае не рекомендуется передавать потенциально большой массив данных в качестве параметра формы подбора. Потенциально большой массив данных, хранимый в форме подбора в параметре типа **ДанныеФормыКоллекция**, может оказаться на клиенте не полным за счет оптимизации работы управляемой формы. Как результат – для передачи параметра будет выполнено дополнительное "дочитывание" данных формы с сервера.

7.2. В целях оптимизации передачи данных между формой объекта и формой подбора рекомендуется использовать временное хранилище, чтение и запись которого должна выполняться на сервере.

Проиллюстрируем методику использования формы подбора на примере подбора элементов справочников **Товары** в табличную часть **Товары** документа **РасходТовара**. (Из демонстрационной конфигурации по платформе **1С:Предприятие**).

Открытие формы подбора из клиентского кода должно приводить не более чем к двум обращениям на сервер. С этой целью локальная команда открытия формы подбора в модуле формы документа **РасходТовара** помещает список товаров из табличной части во временное хранилище (первый вызов) и открывает форму подбора (второй вызов), передавая адрес временного хранилища:

&НаКлиенте  
Процедура ОбработчикКомандыПодбора()  
 АдресТоваровВХранилище = ПоместитьТоварыВХранилище();  
 ПараметрыПодбора = Новый Структура("АдресТоваровДокумента, ВидЦен, Склад", АдресТоваровВХранилище, Объект.ВидЦен, Объект.Склад);  
 ФормаПодбора = ОткрытьФорму("Документ.РасходТовара.Форма.ФормаПодбора", ПараметрыПодбора, ЭтотОбъект);  
КонецПроцедуры

&НаСервере  
// Функция помещает список товаров во временное хранилище и возвращает адрес  
Функция ПоместитьТоварыВХранилище()  
 Возврат ПоместитьВоВременноеХранилище(Объект.Товары.Выгрузить(,"Товар,Цена,Количество"), УникальныйИдентификатор);  
КонецФункции

Форма подбора получает список выбранных товаров из временного хранилища в обработчике **ПриСозданииНаСервере**:

&НаСервере  
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)  
 АдресТоваровДокумента = Параметры.АдресТоваровДокумента;  
 Товары.Загрузить(ПолучитьИзВременногоХранилища(АдресТоваровДокумента));  
 …  
КонецПроцедуры

Закрытие формы подбора должно приводить не более чем к двум обращениям на сервер. При закрытии форма подбора помещает список выбранных товаров во временное хранилище (первый вызов):

&НаКлиенте  
Процедура ОКВыполнить()  
 ЗаписатьПодборВХранилище();  
 ВладелецФормы.ОбработатьПодбор();  
 Закрыть();  
КонецПроцедуры

&НаСервере  
Процедура ЗаписатьПодборВХранилище()  
 ПоместитьВоВременноеХранилище(Товары.Выгрузить(), АдресТоваровДокумента);  
КонецПроцедуры

Затем форма документа восстанавливает список товаров из временного хранилища (второй вызов на сервер):

&НаКлиенте  
Процедура ОбработатьПодбор() Экспорт  
 ПолучитьТоварыИзХранилища(АдресТоваровВХранилище);   
КонецПроцедуры

&НаСервере  
Процедура ПолучитьТоварыИзХранилища(АдресТоваровВХранилище)  
 Объект.Товары.Загрузить(ПолучитьИзВременногоХранилища(АдресТоваровВХранилище));  
КонецПроцедуры

См. также

* [Использование значений, влияющих на поведение клиентского приложения](https://its.1c.ua/db/content/v8std/src/400/i8100459.htm?_=1580137113)
* [Использование объекта ДанныеФормыКоллекция](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100628.htm?_=1580137113)
* [Оптимизация клиент-серверного взаимодействия прикладных решений](http://its.1c.ru/db/metod81#content:4105:1) (ИТС)

## Минимизация кода, выполняемого на клиенте

#std629

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1.1. Необходимо минимизировать объем кода, который выполняется на стороне клиентского приложения. В частности не следует выполнять на клиенте сложные алгоритмы, требующие значительных ресурсов компьютера. В таких случаях выполнение алгоритма на клиенте может занимать гораздо больше времени, чем передача управления с клиента на сервер, выполнение алгоритма на сервере и возврат результата обратно на клиент.

Следует размещать такие алгоритмы в серверном коде, выполняя к ним минимально необходимое число обращений с клиента.

См. также: [*Минимизация количества серверных вызовов*](https://its.1c.ua/db/content/v8std/src/400/i8100487.htm?_=1580137113)

Это требование продиктовано тем, что

* как правило, клиентский компьютер менее производительный, чем серверный компьютер;
* необходимостью приемлемого качества работы в веб-клиенте. Клиентский код выполняется интерпретатором встроенного языка, который в веб-клиенте работает заметно медленнее, чем в тонком или толстом клиенте.

1.2. Рекомендуется оставлять на клиенте такие алгоритмы, скорость работы которых заведомо быстрее, чем затраты, необходимые на вызов одной серверной функции. Например, перерасчет доступности элементов управления в форме при изменении пользователем данных выполняется на клиенте, т.к. [контекстный серверный вызов](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100636.htm?_=1580137113) для сложной формы может сделать работу пользователя в этой форме неприемлемой.

См. также: [*Особенности табличного документа в веб-клиенте*](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/700/i8100533.htm?_=1580137113)

2.1. Исключение из этого правила составляют отдельные случаи, когда функциональная подсистема предназначена для работы с программным обеспечением, установленным на клиентском компьютере. Например, работа с торговым оборудованием, интеграция с клиент-банком, формирование печатных форм в офисные программы и т.п.

2.2. В тех случаях когда функциональная подсистема предназначена для работы с клиентским программным обеспечением только в определенных режимах работы клиента, следует использовать директивы препроцессора. Например, для кода, недоступного в веб-клиенте:

#Если ВебКлиент Тогда  
Предупреждение(НСтр("ru = 'Загрузка адресного классификатора не доступна в веб-клиенте.'"));  
#Иначе  
ОткрытьФорму("РегистрСведений.АдресныйКлассификатор.Форма.ФормаЗагрузкиАдресногоКлассификатора");  
#КонецЕсли

## Доступ к файловой системе из кода конфигурации

#std542

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При обращении из кода конфигурации к файлам и каталогам файловой системы следует иметь в виду, что доступ к ним ограничен правами пользователя операционной системы, от имени которого запущено приложение.

1.1. Если доступ к файловой системе осуществляется из кода, выполняемого на клиенте, то он выполняется под правами пользователя, от имени которого запущено приложение (тонкий, толстый или веб-клиент). Как правило, это текущий пользователь операционной системы.

1.2. Если доступ к файловой системе осуществляется из кода, выполняемого на сервере, то:

* при использовании клиент-серверной информационной базы, доступ ограничен правами пользователя, от имени которого запущен сервер 1С:Предприятия (\*[[4]](#footnote-4));
* при использовании файловой базы, опубликованной на веб-сервере, доступ ограничен правами пользователя, от имени которого запущен веб-сервер.

|  |
| --- |
| Область применения (уточнение): управляемое приложение, обычное приложение.  2. Запрещается выполнять запись каких-либо файлов в каталог исполняемых файлов 1С:Предприятия, получаемого с помощью метода **КаталогПрограммы**. Использование метода **КаталогПрограммы** допустимо только для чтения или запуска файлов. Например, при работе в ОС Windows, для запуска копии тонкого клиента 1С:Предприятия текущей версии, допустимо использовать:  ЗапуститьПриложение(КаталогПрограммы() + "1cv8s.exe"); |

3. Даже если не планируется [локализация конфигурации на другие языки](https://its.1c.ua/db/content/v8std/src/400/i8100761.htm?_=1580137113), следует обеспечивать переносимость файлов, сформированных из кода конфигурации, между различными операционными системами с различными кодировками. Для этого необходимо:

3.1. В именах файлов, автоматически формируемых из кода конфигурации, указывать только английские буквы, а также цифры, а в качестве кодировки текстовых файлов использовать только UTF-8 (именно этот формат предпочтителен, т.к. только с ним корректно работает операционная система macOS).

Это ограничение распространяется на файлы сообщений обмена, выгрузки данных, электронных документов и пр., которые автоматические формируются системой, в том числе на файлы, упакованные в архивы (например, zip). Исключение составляют случаи, когда на формат файлов невозможно повлиять, например, это формат сторонней системы.

3.2. В тех случаях, когда имя файла не генерируется системой, а его явно вводит пользователь, разрешить ввод русскоязычных имен, но при этом дать возможность транслитерировать его в англоязычное имя. По умолчанию, если это технически возможно и не снижает удобство работы, рекомендуется предлагать англоязычное имя файла, а для текстовых файлов – сохранение в кодировке UTF-8.

Также эти рекомендации по выбору имени и кодировки файла следует разместить в справке к тем местам программы, где пользователь имеет возможность сохранять файлы и выбирать кодировку.

В конфигурациях на базе **Библиотеки стандартных подсистем** для транслитерации имен файлов рекомендуется использовать функцию **СтроковыеФункцииКлиентСервер.СтрокаЛатиницей**.

#### Работа с временными файлами и каталогами

При необходимости использования временных файлов и каталогов необходимо соблюдать следующие требования:

1. Для получения имени временного файла следует использовать метод **ПолучитьИмяВременногоФайла** (исключение составляет веб-клиент, см. ниже п. 3). В противном случае возможна некорректная работа конфигурации в многопользовательском режиме, с включенными профилями безопасностями, возникновение проблем с правами доступа к файлам операционной системы, а также неконтролируемое увеличение количества ненужных временных файлов, которые не будут своевременно удалены.

Например, неправильно:

ИмяПромежуточногоФайла = "C:\Временные файлы 1С\TempFile.xml";  
Данные.Записать(ИмяПромежуточногоФайла);

У текущего пользователя может не быть прав на запись в указанный каталог. Кроме того, при одновременном выполнении этого кода из двух разных сеансов возникнет ошибка.

Правильно:

ИмяПромежуточногоФайла = ПолучитьИмяВременногоФайла("xml");  
Данные.Записать(ИмяПромежуточногоФайла);

При использовании этой функции будет получено уникальное имя, гарантирован доступ к файлу.

Кроме того, при использовании метода **ПолучитьИмяВременногоФайла** платформа **1С:Предприятие** сохраняет контроль над такими файлами и автоматически удаляет их при перезапуске рабочего процесса (если файл был создан на стороне сервера) или клиентского приложения (если файл был создан на стороне клиента).

Если же имя временного файла было сформировано каким-то другим способом, и прикладной код не удалил (либо по какой-то причине не смог удалить) ранее созданный временный файл, то платформа такой файл не контролирует, и он остается в файловой системе на неопределенное время. Накапливание «потерянных» временных файлов может представлять серьезную проблему, особенно для информационных баз с большим количеством активно работающих пользователей (например, при работе в режиме сервиса).

Таким образом, неправильно**:**

Каталог = КаталогВременныхФайлов();  
ИмяФайла = Строка(Новый УникальныйИдентификатор) + ".xml";  
ИмяПромежуточногоФайла = Каталог + ИмяФайла;  
Данные.Записать(ИмяПромежуточногоФайла);

Если по каким-то причинам прикладной код не удалит созданный файл (например, между блоками создания и удаления временного файла возникнет штатное или нештатное исключение), этот файл так и останется в каталоге временных файлов.

Правильно**:**

ИмяПромежуточногоФайла = ПолучитьИмяВременногоФайла("xml");  
Данные.Записать(ИмяПромежуточногоФайла);

При использовании метода **ПолучитьИмяВременногоФайла** будет получено уникальное имя, гарантирован доступ к файлу, а также временный файл будет автоматически удален платформой 1С:Предприятие после завершения рабочего процесса сервера или клиентского приложения.

2. Для создания временного каталога рекомендуется также использовать имя, полученное при помощи метода **ПолучитьИмяВременногоФайла**(исключение составляет веб-клиент, см. ниже п. 3). Это гарантирует уникальность имени создаваемого каталога при работе в многопользовательском режиме и гарантирует, что после перезапуска рабочего процесса или клиентского приложения временный каталог будет автоматически удален платформой 1С:Предприятие. После этого, внутри созданного каталога можно создавать другие каталоги и файлы без ограничений.

3.1. При выполнении кода веб-клиентом метод **ПолучитьИмяВременногоФайла** недоступен. Поэтому для формирования имен временных файлов и каталогов необходимо использовать функцию **КаталогВременныхФайлов** и объект **УникальныйИдентификатор**.

Неправильно:

Каталог = КаталогВременныхФайлов();  
ИмяФайла = "TempDataFile.xml";  
ИмяПромежуточногоФайла = Каталог + ИмяФайла;  
Данные.Записать(ИмяПромежуточногоФайла);

Правильно:

Каталог = КаталогВременныхФайлов();  
ИмяФайла = Строка(Новый УникальныйИдентификатор) + ".xml";  
ИмяПромежуточногоФайла = Каталог + ИмяФайла;  
Данные.Записать(ИмяПромежуточногоФайла);

3.2. Если в конфигурацию встроена **Библиотека стандартных подсистем**, для создания временных каталогов на стороне клиента необходимо использовать процедуру **ФайловаяСистемаКлиент.СоздатьВременныйКаталог**.

4. После окончания работы с временным файлом или каталогом, его необходимо удалить самостоятельно. Нельзя рассчитывать на автоматическое удаление файлов и каталогов при следующем запуске платформы, это может привести к исчерпанию свободного места в каталоге временных файлов.

ИмяПромежуточногоФайла = ПолучитьИмяВременногоФайла("xml");  
Данные.Записать(ИмяПромежуточногоФайла);

// Работа с файлом  
...

// Удаляем временный файл  
Попытка  
   УдалитьФайлы(ИмяПромежуточногоФайла);  
Исключение  
   ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие'"), УровеньЖурналаРегистрации.Ошибка, , , ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));  
КонецПопытки;

*См. также:*[*Использование Журнала регистрации.*](https://its.1c.ua/db/content/v8std/src/400/blanki8100498.htm?_=1580137113)

5. При использовании временных файлов и каталогов на сервере, необходимо полностью завершать работу с ними в рамках одного серверного вызова. При работе конфигурации с использованием кластера серверов, при следующем вызове эти файлы могут стать недоступны, так как код начнет исполняться на другом компьютере. При необходимости сохранить данные между серверными вызовами в пределах одного сеанса следует использовать временное хранилище платформы (методы **ПоместитьВоВременноеХранилище**, **ПолучитьИзВременногоХранилища**).

5.1. В редких случаях может возникнуть необходимость передачи данных во временных файлах между сеансами, например, при подготовке данных для фонового задания, при организации длительного процесса, обслуживающего несколько последовательных вызовов web-сервиса. Необходимо самостоятельно обеспечивать гарантировано общее место хранения, права для доступа к файлам из разных мест их обработки, удаление файлов по истечению сроков их обработки или аварийного завершения процесса обработки. Рекомендуется использовать следующий подход:

* Для обеспечения доступа со всех возможных мест обработки заводится константа для хранения общего пути к файлам, доступного для доступа со всех серверов кластера;
* При создании временных файлов их имена заносятся во вспомогательный регистр сведений с сохранением времени создания файла;
* При штатном прохождении процесса, последняя операция, которой были нужны файлы, перед своим завершением удаляет как сам файл, так и записи о них во вспомогательном регистре;
* Вспомогательное регламентное задание периодически проверяет наличие записей во вспомогательном регистре, время существования которых заведомо превышает время штатного завершения процесса. При обнаружении таких записей, задание удаляет временные файлы и записи о них.

#### Передача файлов между клиентом и сервером

1. При одновременной работе с файлом на клиенте и на сервере необходимо использовать передачу файла через временное хранилище (методы **ПоместитьФайлы**, **ПолучитьФайл**, **ПолучитьФайлы**, **НачатьПомещениеФайла**, **ПоместитьВоВременноеХранилище**, **ПолучитьИзВременногоХранилища**). В общем случае клиент и серверы кластера - это разные компьютеры с разной файловой системой, причем доступ к файлам может происходить под разными пользователями ОС с различными правами.

**Неправильно:**

&НаКлиенте  
Процедура ОбработатьФайл()  
   ...  
   ИмяФайла = "C:\Файлы для обработки\Загрузка.xml";  
   Результат = ПроизвестиОбработкуНаСервере(ИмяФайла);  
   ...

КонецПроцедуры

&НаСервере  
Функция ПроизвестиОбработкуНаСервере(ИмяФайла)

   Чтение = Новый ЧтениеТекста(ИмяФайла);  
   ...

   Результат = Чтение.Прочитать();  
   Возврат Результат;

КонецФункции

Правильно:

&НаКлиенте  
Процедура ОбработатьФайл()  
   ...

   ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";  
   ОписаниеОповещения = Новый ОписаниеОповещения(  
      "ОбработатьФайлЗавершение", ЭтотОбъект);

    НачатьПомещениеФайла(ОписаниеОповещения,,  
           ИмяФайлаДляОбработки,  Ложь,  
           УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте  
Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

   ...  
   Результат = ПроизвестиОбработкуНаСервере(Адрес);  
   ...

КонецПроцедуры

&НаСервере  
Функция ПроизвестиОбработкуНаСервере(Адрес)

   Данные = ПолучитьИзВременногоХранилища(Адрес);  
   ИмяПромежуточногоФайла = ПолучитьИмяВременногоФайла("txt");  
   Данные.Записать(ИмяПромежуточногоФайла);

   Чтение = Новый ЧтениеТекста(ИмяПромежуточногоФайла);  
   ...  
   Результат = Чтение.Прочитать();  
   ...

   УдалитьФайлы(ИмяПромежуточногоФайла);

   Возврат Результат;

КонецФункции

2. Для сохранения данных во временном хранилище между несколькими серверными вызовами, при помещении его в хранилище необходимо использовать параметр **УникальныйИдентификаторФормы** метода **ПоместитьФайл**, передав в него идентификатор текущей формы. Такие значения будут удалены из временного хранилища только при закрытии указанной формы. При этом, при повторном помещении того же файла во временное хранилище, предыдущее значение необходимо удалять вручную. Например:

**Неправильно:**

&НаКлиенте  
Процедура ОбработатьФайл()

   ...  
   // Первый серверный вызов  
   ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";  
   ОписаниеОповещения = Новый ОписаниеОповещения(  
      "ОбработатьФайлЗавершение", ЭтотОбъект);

   НачатьПомещениеФайла(ОписаниеОповещения,,  
      ИмяФайлаДляОбработки, Ложь,  
      УникальныйИдентификатор);

   ...

КонецПроцедуры

&НаКлиенте  
Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

   ...  
   Результат = ПроизвестиНачальнуюОбработкуНаСервере(Адрес);  
   ПродолжитьОбработкуФайла();  
   ...

КонецПроцедуры

&НаКлиенте  
Процедура ПродолжитьОбработкуФайла()

   ...  
   // Второй серверный вызов с той же версией файла  
   Результат = ПроизвестиПромежуточнуюОбработкуНаСервере(Адрес);  
   ...

   // Третий серверный вызов с новой версией файла  
   ОписаниеОповещения = Новый ОписаниеОповещения(  
   "ПродолжитьОбработкуФайлаЗавершение", ЭтотОбъект);

   НачатьПомещениеФайла(ОписаниеОповещения,,  
      ИмяФайлаДляОбработки, Ложь,  
      УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте

Процедура ПродолжитьОбработкуФайлаЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

   ...  
   Результат = ПроизвестиКонечнуюбОбработкуНаСервере(Адрес);  
   ...

КонецПроцедуры

При этом во временном хранилище формы останется две копии файлов. Адрес второй копии будет находиться в переменной **Адрес**, а адрес первой копии будет утерян. Это приводит к затрате дополнительных ресурсов приложения, замедлению работы.

**Правильно:**

&НаКлиенте  
Процедура ОбработатьФайл()

   ...  
   // Первый серверный вызов  
   ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";

   ОписаниеОповещения = Новый ОписаниеОповещения(  
      "ОбработатьФайлЗавершение", ЭтотОбъект);

   НачатьПомещениеФайла(ОписаниеОповещения,,  
      ИмяФайлаДляОбработки, Ложь,  
      УникальныйИдентификатор);  
   ...

КонецПроцедуры

&НаКлиенте  
Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

   ...  
   Результат = ПроизвестиНачальнуюОбработкуНаСервере(Адрес);  
   ПродолжитьОбработкуФайла();  
   ...

КонецПроцедуры

&НаКлиенте  
Процедура ПродолжитьОбработкуФайла()

   ...  
   // Второй серверный вызов с той же версией файла  
   Результат = ПроизвестиПромежуточнуюОбработкуНаСервере(Адрес);  
   ...

   // Третий серверный вызов с новой версией файла  
   УдалитьИзВременногоХранилища(Адрес);

   ОписаниеОповещения = Новый ОписаниеОповещения(  
   "ПродолжитьОбработкуФайлаЗавершение", ЭтотОбъект);

   НачатьПомещениеФайла(ОписаниеОповещения,,  
      ИмяФайлаДляОбработки, Ложь,  
      УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте  
Процедура ПродолжитьОбработкуФайлаЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

   ...  
   Результат = ПроизвестиКонечнуюбОбработкуНаСервере(Адрес);  
   ...

КонецПроцедуры

3. Если в конфигурацию встроена **Библиотека стандартных подсистем** для помещения файлов во временное хранилище необходимо использовать процедуры **ЗагрузитьФайл** и **ЗагрузитьФайлы** общего модуля **ФайловаяСистемаКлиент**. Для сохранения данных файла между несколькими серверными вызовами необходимо использовать свойство **ИдентификаторФормы** параметра **ПараметрыЗагрузки**:

&НаКлиенте  
Процедура ОбработатьФайл()  
   ...  
   ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";  
   ОписаниеОповещения = Новый ОписаниеОповещения("ОбработатьФайлЗавершение", ЭтотОбъект);  
  
   ПараметрыЗагрузки = ФайловаяСистемаКлиент.ПараметрыЗагрузкиФайла();  
   ПараметрыЗагрузки.ИдентификаторФормы = УникальныйИдентификатор;  
   ПараметрыЗагрузки.Интерактивно = Ложь;

   ФайловаяСистемаКлиент.ЗагрузитьФайл(ОписаниеОповещения,  
   ПараметрыЗагрузки, ИмяФайлаДляОбработки);  
  
КонецПроцедуры

&НаКлиенте  
Процедура ОбработатьФайлЗавершение(ПомещенныйФайл, ДополнительныеПараметры)  
  
   ...  
   Результат = ПроизвестиОбработкуНаСервере(Адрес);  
   ...  
  
КонецПроцедуры

См. также

* [Установка внешних компонент и расширений платформы](https://its.1c.ua/db/content/v8std/src/1%C2%A0100/900/i8100700.htm?_=1580137113)
* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1580137113)

## Оптимизация использования оперативной памяти

#std725 Методическая рекомендация (полезный совет)

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Не следует разрабатывать решения исходя из неограниченного объема оперативной памяти.  Для многопользовательских систем любое неэффективное использование памяти может катастрофически сказаться на работоспособности.

Следует избегать формирования больших структур данных в памяти. Если объём данных, с которыми работает бизнес-логика, сам по себе ничем не ограничен, его нужно ограничивать искусственно, обрабатывая данные порциями и сохраняя результаты в базу или файлы.

2. При потенциально неограниченных выборках данных из ИБ следует получать данные из базы порциями фиксированного размера.  
Например, неправильно:

 Запрос = Новый Запрос;  
 Запрос.Текст =  
  "ВЫБРАТЬ  
  | Номенклатура.Ссылка,  
  | Номенклатура.Наименование,  
  | Номенклатура.ВидНоменклатуры  
  |ИЗ  
  | Справочник.Номенклатура КАК Номенклатура";

 // Выгрузка всего справочника в таблицу значений  
 Номенклатура = Запрос.Выполнить().Выгрузить();   
 Для каждого ПозицияНоменклатуры Из Номенклатура Цикл  
  // Обработка элемента справочника  
  // ...  
 КонецЦикла;

поскольку весь результат запроса сразу помещается в память, в таблицу значений.  
Также неправильно:

 Запрос = Новый Запрос;  
 Запрос.Текст =  
  "ВЫБРАТЬ  
  | Номенклатура.Ссылка,  
  | Номенклатура.Наименование,  
  | Номенклатура.ВидНоменклатуры  
  |ИЗ  
  | Справочник.Номенклатура КАК Номенклатура";

 РезультатЗапроса = Запрос.Выполнить();  
 // Обход результата запроса  
 ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();  
 Пока ВыборкаДетальныеЗаписи.Следующий() Цикл  
  // Обработка элемента выборки  
  // ...  
 КонецЦикла;

поскольку и в этом случае при выполнении запроса его результат будет сначала считан в память целиком ([[5]](#footnote-5)).

Правильно ограничивать результат запроса искусственно:

ВсеОбработано = Ложь;  
Пока Истина Цикл  
 Запрос = Новый Запрос;  
 Запрос.Текст =  
  "ВЫБРАТЬ ПЕРВЫЕ 1000  
  | Номенклатура.Ссылка,  
  | Номенклатура.Наименование,  
  | Номенклатура.ВидНоменклатуры  
  |ИЗ  
  | Справочник.Номенклатура КАК Номенклатура  
  |ГДЕ  
  | <условие выборки необработанных записей>";

 РезультатЗапроса = Запрос.Выполнить();  
 ВсеОбработано = РезультатЗапроса.Пустой();  
 Если ВсеОбработано Тогда  
  Прервать;  
 КонецЕсли;

 // Обход порции результата запроса  
 ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();  
 Пока ВыборкаДетальныеЗаписи.Следующий() Цикл  
  // Обработка элемента выборки  
  // ...  
 КонецЦикла;

КонецЦикла;

Также правильно:

 Выборка = Справочники.Номенклатура.Выбрать(..., Отбор);  
 Пока Выборка.Следующий() Цикл  
  // Обработка элемента выборки  
  // ...  
 КонецЦикла;

поскольку в этом случае платформа 1С:Предприятие выполняет получение данных из базы порциями фиксированного размера.

Кроме того, число элементов выборки автоматически ограничивает платформа 1С:Предприятие в запросах динамических списков.

3. Недопустимо работать с большими XML документами с помощью объектов встроенного языка, предназначенных для обработки файлов целиком: текстовые документы в **ТекстовыйДокумент**, XML в **ДокументDOM** и HTML в **ДокументHTML**, а также создавать в памяти XDTO-пакеты размером с весь XML-файл целиком.

В противном случае, весь файл загружается в оперативную память целиком. Исключения составляют отдельные случаи, когда необходим произвольный доступ к содержимому файла, к какой-то конкретной его части.

Следует использовать объекты для последовательной записи и последовательного чтения: **ЧтениеXML**, **ЧтениеТекста**, **ЗаписьXML**, **ЗаписьТекста**, с помощью которых можно прочитать файл порциями и расходовать память экономно.

При использовании механизмов XDTO неправильно зачитывать в память весь XML-файл целиком (**ФабрикаXTDO.ПрочитатьXML(ЧтениеXML)**). Вместо этого следует зачитывать XML-файл последовательно, с помощью объекта **ЧтениеXML**, а его отдельные фрагменты (теги) десериализовывать с помощью фабрики XDTO.

4. Другая распространенная причина неэффективное использование памяти - утечки памяти. К утечкам памяти приводит создание циклических ссылок – память выделяется и не освобождается. Например, если есть объекты, внутри которых вложены другие объекты, и где-то в глубине они ссылаются на самый верхний объект. В результате образуется циклическая ссылка.  
Упрощенный пример циклической ссылки:

Данные = Новый Структура;  
Данные.Вставить("Ключ", Данные);

Следует разрывать (очищать) ссылки, когда объект становится не нужен.  
Например, для примера выше:

Данные.Ключ = Неопределено;

Для выявления утечек памяти можно применять технологический журнал, включив в файл настройки параметров технологического журнала logcfg.xml элемент <leaks>.

Подробнее см.:

* [Документация к платформе 1С:Предприятие, Приложение 3. Описание и расположение служебных файлов - logcfg.xml](http://its.1c.ru/db/v83doc#content:27:1)
* [Поиск циклических ссылок](https://its.1c.ru/db/metod8dev#content:5859)

5. Чрезмерное (неоправданное) применение [общих модулей с повторным использованием возвращаемых значений  может также приводить к излишнему потреблению памяти](https://its.1c.ua/db/content/v8std/src/400/i8100724.htm?_=1580137113).

## Таймауты при работе с внешними ресурсами

#std748

Область применения: управляемое приложение, обычное приложение.

1. При работе с внешними ресурсами с помощью объектов **WSОпределения**, **WSПрокси**, **HTTPСоединение**, **FTPСоединение**, **ИнтернетПочтовыйПрофиль** следует задавать таймаут – предельное время  ожидания выполнения операции. В противном случае, в результате бесконечного ожидания программа зависнет или часть функционала программы станет недоступна.

Установка таймаута является защитой от целого ряда внешних факторов:

* нестабильного подключения к Интернету, когда регулярно происходит прерывание связи, и система не может получить цельный ответ сервера, к которому выполняется подключение;
* при включенных антивирусных программах или при неправильных настройках брандмауэра;
* неправильной настройки прокси-сервера;
* ненадежной работы веб-сервера из-за возросшей нагрузки или некорректной работы скриптов.

Например, при получении описания веб-сервиса и вызове его операций – если удаленная сторона долго не отвечает (например, выключена, находится на обслуживании или возникли временные неполадки), ожидание ответа может длиться бесконечно. Поэтому если веб-сервис был вызван в результате интерактивных действий пользователя, то внешне будет выглядеть так, что «программа зависла»; а если веб-сервис вызывается из регламентного задания, то связанная с ним часть функционала программы может стать недоступна.

2. В общем виде, время выполнения операции с внешними ресурсами складывается из шести этапов:

* DNS Lookup — время, потраченное на определение IP адреса по доменному имени (если применимо);
* Connect — установка соединения с веб-сервером по полученному IP-адресу;
* Send — отправка данных на веб-сервер;
* Wait — ждем, пока данные дойдут до веб-сервера и он их обработает;
* Receive — получение ответа от веб-сервера;
* Cache Read – получение данных от веб-сервера.

Например, при таймауте в 60 секунд программа и вызываемый внешний ресурс должны успеть выполнить шесть выше перечисленных этапов операции, иначе соединение будет разорвано, а передача данных прервана. Однако если в процессе выполнения операции возникнет сбой, то система и/или пользователь будет зря ожидать 60 секунд.

Поэтому величину таймаута рекомендуется определять, исходя из ожидаемого времени выполнения конкретной операции:

* Для быстрых операций (например, проверка доступности сервера) величина таймаута должна выбираться, соответственно, небольшой;
* В общем случае, не следует выбирать таймаут более 3 минут, чтобы при недоступности удаленной стороны не допустить эффект «зависания» программы;
* Но если операция выполняется долго из-за этапов Send или Cache Read, т.е. это передача больших объемов данных на веб-сервер или загрузка большого файла с внешнего ресурса, то следует устанавливать большой таймаут, исходя из оценки объема передаваемых данных, но не более 12 часов.

Подобнее о рекомендуемых величинах таймаута для различных операций см. в таблице п. 4.

3. Рекомендации по снижению величин таймаута и повышению отзывчивости программы при работе с внешними ресурсами.

3.1. При разработке веб-сервисов, на операции которых предусмотрен таймаут более 20 секунд (ориентировочно), рекомендуется:

* предусмотреть в веб-сервисе отдельную контрольную операцию Ping;
* при работе с этим веб-сервисом, предварительно получать для нее прокси с небольшим таймаутом в 7 секунд и вызывать контрольную операцию Ping;
* только после этого получать основной прокси.

Пример вызова веб-сервиса.

|  |  |
| --- | --- |
| **Неправильно** | **Правильно** |
| Реализация модуля веб-сервиса PingPong: | |
| Функция Pong(Знач Параметр)   Возврат СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru = 'Привет, %1'"), Параметр); КонецФункции | Функция Ping()   Возврат Истина; // Проверка связи КонецФункции  Функция Pong(Знач Параметр)   Возврат СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru = 'Привет, %1'"), Параметр); КонецФункции |
| Реализация вызывающей стороны (без использования Библиотеки стандартных подсистем): | |
| // Ждем не более минуты PingPong = Новый WSПрокси(АдресВебСервиса, , , , , 60); Результат = PingPong.Pong(НСтр("ru = 'Мяч'")); | // Ждем не более 3 секунд PingPong = Новый WSПрокси(АдресВебСервиса, , , , , 3); PingPong.Ping(); // проверка связи  // Сервис жив, далее работаем с ним и ждем не более минуты PingPong = Новый WSПрокси(АдресВебСервиса, , , , , 60);  Результат = PingPong.Pong(НСтр("ru = 'Мяч'")); |

При использовании Библиотеки стандартных подсистем:

* для работы с веб-сервисами предназначена функция WSПрокси общего модуля ОбщегоНазначения (включает в себя поддержку контрольной операции **Ping**);
* для получения данных по протоколам HTTP(S) и FTP(S) – подсистема «Получение файлов через Интернет».

Пример реализации вызывающей стороны с использованием Библиотеки стандартных подсистем:

// Сделать контрольный вызов Ping и ждать не более минуты на дальнейших операциях.  
PingPong = ОбщегоНазначения.WSПрокси(АдресВебСервиса,..., 60, Истина);  
// Сервис точно жив, далее работаем с ним.  
Результат = PingPong.Pong(НСтр("ru = 'Мяч'"));

3.2. Для других видов внешних ресурсов (не веб-сервисов) рекомендуется применять аналоги операции **Ping**. Например:

* для сервисов, работающих через REST API – это контрольная отправка тестовой команды; в большинстве случаев, если ответ с кодом 200, то сервис работает;
* для FTP/WebDAV-ресурсов – это контрольная загрузка (отправка) файла-пустышки.

3.3. Веб-сервисы, операции, которых занимают объективно много времени из-за этапа Wait (т.е. долго отрабатывает само веб-приложение), и они не могут быть ускорены (оптимизированы) по объективным причинам, следует переводить на асинхронный режим выполнения:

* запускать фоновое задание для выполнения подобной «тяжелой» операции,
* и предусмотреть дополнительные операции по проверке готовности и получению результата.

Пример асинхронного вызова веб-сервиса.

|  |  |
| --- | --- |
| **Неправильно** | **Правильно** |
| Реализация модуля веб-сервиса Long: | |
| Функция GetData()     Результат = <очень длительные вычисления>;     Возврат Результат; КонецФункции | Функция StartDoLong()     // запуск фонового задания     ИдентификаторОперации = ...     // возвращаем идентификатор операции для отслеживания ее готовности     Возврат ИдентификаторОперации; КонецФункции  Функция IsReady(Знач ИдентификаторОперации)     // проверяем, завершено ли фоновое задание по переданному идентификатору     Готовность = ...     Возврат Готовность; КонецФункции  Функция GetData(Знач ИдентификаторОперации)     Результат = <получаем уже готовый результат по переданному идентификатору >;     Возврат Результат; КонецФункции |
| Реализация вызывающей стороны: | |
| Long = Новый WSПрокси(АдресВебСервиса, , , , , 600); // ждем 1 час  Результат = Long.GetData(); | Long = Новый WSПрокси(АдресВебСервиса, , , , , 600); // ждем 1 час  ИдентификаторОперации = Long.StartDoLong();  Пока Не Long.IsReady(ИдентификаторОперации) Цикл     <ждем определенный интервал времени> КонецЦикла; Результат = Long.GetData(ИдентификаторОперации);  \* это лишь упрощенная схема реализации вызывающей стороны; в действительности, код вызывающей стороны также должен быть реализован асинхронно с помощью включения регламентного задания, либо периодического обработчика ожидания на клиенте, который проверяет готовность и получает результат. |

4. Рекомендуемые величины таймаутов для различных операций:

|  |  |
| --- | --- |
| **Операция** | **Таймаут (секунд)** |
| Получение описания веб-сервиса | 7 |
| Проверка корректности введенного адреса, взаимодействие с менеджером сервиса в модели сервиса и прочие «быстрые» операции | 10-20 |
| Получение сведений об одном контрагенте, обмен сообщениями, отправка SMS, удаленное администрирование ИБ в модели сервиса | 60-1201 |
| Передача сообщений обмена данными через веб-сервис или получение файлов из внешнего ресурса до 1 Мб. | 120-1801 |
| Загрузка файлов более 1 Мб | Если известен размер файла, то размер в мегабайтах \* 1282, иначе предельное время загрузки, но не более 43200 3 |

1 Следует вызывать только после контрольной операции **Ping**.

2 Загрузка 1 мегабайта данных занимает 128 секунд, при скорости 64 кбит/с, т.к. сотовые операторы в определенных случаях ограничивают скорость загрузки этой величиной.

3 Таймаут продолжительностью 43200(12 часов) сек. является компромиссным решением, т.к. в случае нештатной ситуации процесс «отвиснет» на следующее утро и вернет управление, в отличие от полностью зависнувшей программы при неустановленном таймауте.

# [Общие вопросы безопасности](https://its.1c.ua/db/v8std/browse/13/-1/36)

## Безопасность прикладного программного интерфейса сервера

#std678

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

При работе в режиме управляемого приложения, клиентское приложение (тонкий или веб-клиент) обращается к серверу 1С:Предприятия посредством открытого HTTP-протокола. Таким образом, сервер 1С:Предприятия может быть вызван извне сторонними программами тем же способом, как это штатно делает клиентское приложение, и злоумышленник может получить несанкционированный доступ к пользовательским данным, нарушить работоспособность сервера.

1. Несанкционированный вызов серверного кода конфигурации с клиента.

1.1. Потенциальную угрозу безопасности представляют все серверные процедуры и функции, доступные для вызова из клиентского кода. Они составляют прикладной программный интерфейс сервера 1С:Предприятия. К ним, как правило, относятся:

* Экспортные процедуры и функции, размещенные в общих модулях с признаками **"Сервер"** и **"Вызов сервера"**. Вызов таких процедур и функций возможен напрямую с клиента.
* *Подробнее см.*[*Ограничение на установку признака****"Вызов сервера"****у общих модулей*](https://its.1c.ua/db/content/v8std/src/500/i8100679.htm?_=1580137113)
* Все процедуры и функции модулей форм объектов с директивами компиляции **&НаСервере**, **&НаСервереБезКонтекста**. Вызов таких процедур и функций доступен из контекста клиента после успешного получения формы, даже если эти процедуры и функции не экспортные. Это делает возможным вызов кода в контексте, который не предполагался разработчиком.  
    
  Например, код модуля формы **Справочник.Сотрудники.ФормаЭлемента:**

&НаКлиенте

Процедура УволитьСотрудника(Команда)

Если ДатаРегистрацииУвольнения > ДатаЗапрета Тогда

ЗарегистрироватьУвольнение();

КонецЕсли;

КонецПроцедуры

&НаСервере

Процедура ЗарегистрироватьУвольнение()

...

КонецПроцедуры

Пример стороннего кода, вызывающий напрямую серверную процедуру для обхода проверки, предусмотренной разработчиком формы в обработке команды **УволитьСотрудника**:

ПараметрыФормы = Новый Структура("Ключ", ВыбранныйСотрудник)

Форма = ПолучитьФорму("Справочник.Сотрудники.ФормаЭлемента", ПараметрыФормы);

Форма.ЗарегистрироватьУвольнение();

1.2. В общем случае не рекомендуется размещать в серверных процедурах и функциях модулей форм код, обеспечивающий бизнес-логику, и который не относится к клиент-серверному взаимодействию и обработке реквизитов формы.

1.3. Особого внимания требуют серверные процедуры и функции, использующие [установку привилегированного режима](https://its.1c.ua/db/content/v8std/src/600/i8100485.htm?_=1580137113), или размещенные в общих модулях с признаком **Привилегированный**.

2. Проникновение небезопасного кода на сервер и его выполнение.

Любые возможности конфигурации по выполнению "внешнего" кода или произвольных текстов запросов на сервере, не являющихся частью самого прикладного решения, представляют серьезную опасность.

Также опасны внешние отчеты и обработки, COM-объекты и внешние компоненты. В частности, код внешних обработок может непосредственно обращаться ко всем общим модулям без признака **"Вызов сервера"**, к модулям объектов и менеджеров объектов конфигурации, пытаться переходить в привилегированный режим.

Такие возможности создают прямую угрозу работоспособности сервера из-за некорректного или преднамеренно вредоносного кода: порчу или похищение данных, зависание или остановка рабочего процесса из-за зацикливания, утечек памяти, ресурсоемких операций и запросов и т.д.

*Подробнее см.*[*Ограничение на выполнение "внешнего" кода.*](https://its.1c.ua/db/content/v8std/src/500/i8100669.htm?_=1580137113)

3. Клиентское приложение (тонкий клиент или веб-браузер) не гарантирует безопасность данных, переданных на сторону клиента. Эти данные легко могут быть перехвачены и прочитаны вредоносным программным обеспечением, установленным на клиентском компьютере.

Серверные процедуры и функции должны возвращать в форму только окончательный результат расчета. Следует избегать передачи в форму исходных или промежуточных данных, которые могут раскрывать побочную, возможно приватную информацию бизнес-процесса.

## Ограничение на установку признака «Вызов сервера» у общих модулей

#std679

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. Не следует всем общим модулям с признаком **Сервер** принудительно устанавливать флажок **Вызов сервера**. В таких общих модулях следует размещать только те процедуры и функции, которые действительно предназначены для вызова из клиентского кода и гарантируют выполнение только тех действий (и передачи только тех данных на сторону клиента), которые разрешены пользователю при его работе в программе. Например, серверная функция, реализующая некоторый алгоритм расчета, должна передавать на сторону клиента окончательный результат этого расчета, но не исходные (или промежуточные) данные для расчета, которые сами по себе могут быть недоступны текущему пользователю.

См. также: [Безопасность прикладного программного интерфейса сервера](https://its.1c.ua/db/content/v8std/src/500/i8100678.htm?_=1580137113), [*Правила создания общих модулей*](https://its.1c.ua/db/content/v8std/src/100/100/i8100469.htm?_=1580137113)

Особого внимания требуют процедуры и функции, использующие установку [привилегированного режима](https://its.1c.ua/db/content/v8std/src/600/i8100485.htm?_=1580137113), или размещенные в общих модулях с признаком **Привилегированный**.

Серверные процедуры и функции, не предназначенные для вызова из клиентского кода, следует размещать в общих модулях без признака **Вызов сервера**.

2.1. Как правило, при разработке объектов конфигурации (справочников, документов и пр.) исходят из того, что в управляемом режиме работа с экземплярами этих объектов (**СправочникОбъект**, **ДокументОбъект** и т.д.) выполняется на стороне сервера. Поэтому в управляемом режиме не гарантируется возможность работы с ними на стороне клиентского приложения.

В частности, в толстом клиенте в режиме управляемого приложения не следует создавать или получать объекты:

* в клиентских общих модулях (пользуясь инструкцией препроцессора **ТолстыйКлиентУправляемоеПриложение**);
* в обычных формах при запуске в управляемом режиме. Такие формы следует использовать только в режиме обычного приложения.

Это позволит избежать выполнения кода модулей объектов и подписок на события на клиенте, а также избыточных серверных вызовов процедур и функций общих модулей из этого кода.

См. также: [*Поддержка толстого клиента, управляемое приложение, клиент-сервер*](https://its.1c.ua/db/content/v8std/src/100/100/i8100680.htm?_=1580137113)

2.2. Если конфигурация **не** рассчитана на работу в толстом клиенте, управляемое приложение, следует снять флажок **Толстый клиент (управляемое приложение, режим клиент-сервер)**, для того чтобы при проверке конфигурации избежать ложных сообщений об ошибках.

## Безопасное хранение паролей

#std740

Область применения: управляемое приложение, мобильное приложение, обычное приложение.

1. При разработке подсистем, взаимодействующих с различными внешними ресурсами (электронной почтой, веб-сервисами, FTP-ресурсами и т.п.) возникает необходимость запрашивать и передавать данные аутентификации к этим ресурсам: логин и пароль.

2. Для сведения к минимуму возможности перехвата пароля злоумышленниками не следует хранить пароли и другую конфиденциальную информацию в информационной базе. При этом минимальный уровень защищенности – в файловых информационных базах, в которых файл базы может быть скопирован целиком любым пользователем информационной базы. В клиент-серверной информационной базе доступ к базе данных, как правило, имеется только у администраторов СУБД.

Таким образом, следует запрашивать логин и пароль у пользователя и передавать их сразу, не сохраняя в информационной базе.

3. В ряде случаев такая схема работы доставляет объективные неудобства или принципиально невозможна:

* интерактивный запрос логина и пароля на каждую операцию может создавать значительный дискомфорт от работы, а временного сохранения на стороне клиента недостаточно;
* взаимодействие с различными внешними ресурсами должно выполняться на сервере, не зависимо от интенсивности работы пользователей с программой.

В таких случаях допустимо организовать хранение паролей и другой конфиденциальной информации в информационной базе, предупредив пользователей о последствиях. Следует помнить, что подобное хранение паролей не решает всех проблем безопасности, а лишь усложняет задачу для злоумышленника.

3.1. При этом не следует хранить пароли и другую конфиденциальную информацию в реквизитах тех же объектов метаданных, с которыми ведется повседневная работа. Для хранения такой информации следует использовать отдельный объект метаданных (например, регистр сведений), организовав к нему безопасный доступ на уровне системы прав доступа 1С:Предприятия.

3.2. При использовании **Библиотеки стандартных подсистем (БСП)** следует использовать безопасное хранилище паролей, которое решает ряд задач:

* Имея доступ к объекту метаданных, пользователь может прочитать содержимое реквизита с паролем, что невозможно при использовании безопасного хранилища. Для исключения случаев несанкционированного доступа к безопасному хранилищу получение и запись данных (паролей) возможна только в привилегированном режиме.
* Данные в безопасном хранилище хранятся в закрытом виде и тем самым исключаются случаи непредумышленной «засветки» паролей.
* Безопасное хранилище исключено из планов обмена, что предотвращает утечку паролей из информационной базы при обмене данными.

Для работы с безопасным хранилищем паролей предназначены процедуры и функции общего модуля **ОбщегоНазначения**: **ЗаписатьДанныеВБезопасноеХранилище**, **ПрочитатьДанныеИзБезопасногоХранилища** и **УдалитьДанныеИзБезопасногоХранилища**. Подробнее см. комментарии к этим функциям в БСП и раздел «3.4. Базовая функциональность - Использование при разработке конфигурации - Безопасное хранилище паролей» документации БСП.

3.3. Не следует хранить пароли в реквизитах формы, их следует извлекать только на стороне сервера и непосредственно перед их использованием. В противном случае, при открытии формы с маскированным вводом (или просмотром) пароля, пароль передается с  сервера на клиент в открытом виде, что делает возможным его перехват. Установка привилегированного режима производится непосредственно перед вызовом функций, а не внутри них, что бы исключить получение или запись любых паролей в сеансе с любыми правами. Безопасность вызова должен обеспечивать вызывающий код, который обращается к конкретным паролям.

Для маскировки пароля на форме в обработчике событии формы **ПриСозданииНаСервере** необходимо разместить следующий код:

 УстановитьПривилегированныйРежим(Истина);  
 Пароли = ОбщегоНазначения.ПрочитатьДанныеИзБезопасногоХранилища(Объект.Ссылка, "Пароль, ПарольSMTP"); // Пароль, ПарольSMTP – ключи соответствия данных в безопасном хранилище  
 УстановитьПривилегированныйРежим(Ложь);

 Пароль = ?(ЗначениеЗаполнено(Пароли.Пароль), ЭтотОбъект.УникальныйИдентификатор, "");  
 ПарольSMTP = ?(ЗначениеЗаполнено(Пароли.ПарольSMTP), ЭтотОбъект.УникальныйИдентификатор, "");

В обработчике события формы **ПриЗаписиНаСервере**:

 Если ПарольИзменен Тогда  
  УстановитьПривилегированныйРежим(Истина);  
  ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, Пароль);  
  УстановитьПривилегированныйРежим(Ложь);  
  Пароль = ?(ЗначениеЗаполнено(Пароль), ЭтотОбъект.УникальныйИдентификатор, "");  
 КонецЕсли;  
   
 Если ПарольSMTPИзменен Тогда  
  УстановитьПривилегированныйРежим(Истина);  
  ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, ПарольSMTP, "ПарольSMTP");  
  УстановитьПривилегированныйРежим(Ложь);  
  ПарольSMTP = ?(ЗначениеЗаполнено(ПарольSMTP), ЭтотОбъект.УникальныйИдентификатор, "");  
 КонецЕсли;

где **Пароль** и **ПарольSMTP** - реквизиты формы. Если пароль ранее был сохранен в программе, то следует присвоить соответствующему реквизиту уникальный идентификатор формы, эмулирующий наличие пароля. При записи объекта в форме, если был введен новый пароль, то записываем его в объект, а реквизит формы вновь затираем уникальным идентификатором.

## Ограничение на выполнение «внешнего» кода

#std669

Область применения: управляемое приложение, обычное приложение.

Помимо программного кода конфигурации, в прикладном решении может исполняться сторонний программный код, который может быть подключен с помощью внешних отчетов, внешних обработок, расширений конфигурации, внешних компонент или другими способами (далее – внешний код). При этом злоумышленник может предусмотреть в нем различные деструктивные действия (как в самом внешнем коде, так и опосредовано, через запуск внешних приложений, внешних компонент, COM-объектов), которые могут нанести вред компьютерам пользователей, серверным компьютерам, а также данным в программе. Пример такой уязвимости: <https://1c.ru/news/info.jsp?id=21537>

Перечисленные проблемы безопасности особенно критичны при работе конфигураций в [модели сервиса](http://v8.1c.ru/overview/Term_000000823.htm). Например, получив доступ к сервису, вредоносный код может получить доступ сразу ко всем приложениям всех пользователей сервиса.

1. Для прикладных решений запрещено выполнение в небезопасном режиме любого кода на сервере 1С:Предприятия, который не является частью самого прикладного решения (конфигурации). Ограничение не распространяется на код, прошедший аудит, и на код, выполняемый на клиенте.

Примеры недопустимого выполнения «внешнего» кода в небезопасном режиме:

* внешние отчеты и обработки (печатные формы и т.п.), расширения конфигурации, внешние компоненты и любые другие аналогичные возможности, с помощью которых пользователи подключают к конфигурации внешний код;
* алгоритмы на встроенном языке, тексты запросов или их фрагменты, которые пользователи интерактивно вводят в режиме 1С:Предприятия, и которые затем передаются в методы глобального контекста **Выполнить** или **Вычислить** (см. «[Ограничения на использование Выполнить и Вычислить на сервере](https://its.1c.ua/db/content/v8std/src/500/i8100770.htm?_=1580137113)»);
* изменение пользователями схем компоновки данных в отчетах, в которых разрешено использование внешних функций (эта возможность закрыта при использовании стандартной формы отчета: она не позволяет пользователям изменять схему компоновки  данных, а из пользовательских полей использовать  функции общих модулей нельзя). В том числе, возможность загрузки пользователями схем компоновки данных из внешних файлов.

При использовании в конфигурации Библиотеки стандартных подсистем (БСП) внешний код допустимо подключать только через соответствующие подсистемы БСП:

* как расширения конфигурации – с помощью средств подсистемы «Базовая функциональность»;
* как внешние отчеты и обработки – через «Дополнительные отчеты и обработки»;
* в виде внешних компонент – через подсистему «Внешние компоненты»;
* для запуска внешних программ – см. [Безопасность запуска приложений](https://its.1c.ua/db/content/v8std/src/500/i8100774.htm?_=1580137113).

При этом указанное в этом пункте требование будет выполнено.

2. По умолчанию, в конфигурации для всех категорий пользователей должна быть отключена возможность интерактивно открывать внешние отчеты и обработки через меню Файл – Открыть. См. пп. 2.2 и 2.3 [Стандартные роли](https://its.1c.ua/db/content/v8std/src/600/i8100488.htm?_=1580137113).  
При этом в настройках программы должна быть предусмотрена обратная возможность разрешить это действие. В случае если администратор разрешает интерактивно открывать внешние отчеты и обработки, то информировать его и пользователей о том, что при открытии файлов внешних отчетов и обработок следует обращать особое внимание на их источник и не открывать файлы, полученные из источников, с которыми нет договоренности о разработке таких отчетов и обработок.

При использовании в конфигурации **Библиотеки стандартных подсистем** отключение интерактивного открытия внешних отчетов и обработок, настройка, а также соответствующие предупреждения уже предусмотрены.

3. Предупреждать администраторов об опасности перед подключением любого внешнего кода.

3.1. Выводимая информация должна включать в себя в явном виде сведения, что внешний код, полученный из недостоверных источников (с которыми, например, нет договоренности о разработке такого кода), может нанести вред компьютерам пользователей, серверным компьютерам, а также данным в программе. При этом администратор должен иметь возможность отказаться от загрузки внешнего кода (а также возможно повторить его загрузку позднее после проведения соответствующего аудита).

При использовании в конфигурации Библиотеки стандартных подсистем такие предупреждения для администратора уже предусмотрены в соответствующих подсистемах.

3.2. В то же время, остальные пользователи программы не должны получать дополнительных предупреждений при исполнении внешнего кода, подключение которого ранее было явно подтверждено администратором.  
Для программного отключения см. раздел 7.10.2. [Отключение механизма защиты от опасных действий](https://its.1c.ua/db/v83doc/bookmark/dev/TI000001873) в документации к платформе 1С:Предприятие.

При использовании в конфигурации **Библиотеки стандартных подсистем**

* подобное отключение предупреждений уже предусмотрено в соответствующих подсистемах;
* запрещено отключать предупреждения об опасных действиях во всех остальных случаях.

4. Если в конфигурации предусмотрены средства обновления конфигурации (из файлов .cf, .cfu), восстановления из резервной копии или загрузки из dt-файла в режиме 1С:Предприятия, то эти операции должны выполняться с соблюдением следующих правил:

* обновление должно быть доступно только пользователю с ролью «Администратор системы»;
* такое обновление должно выполняться только интерактивно текущим пользователем, а не служебным пользователем с полными правами;
* перед обновлением конфигурации из файла или восстановления из резервной копии, администратору должно показываться предупреждение о том, что он должен убедиться, что файл обновления получен из надежного источника;
* при обновлении конфигурации через Интернет, должно использоваться защищенное соединение (см. п. 7) и надежный источник, о чем нужно предупредить пользователя, когда он настраивает параметры подключения к источнику обновления.

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) операции обновления конфигурации и восстановления из резервной копии следует выполнять только средствами подсистем «Обновление конфигурации» и «Резервное копирование ИБ» БСП. При этом автоматически будут выполнены все требования, перечисленные выше в этом пункте.

5. Если в конфигурации предусмотрены средства загрузки произвольных файлов в программу, то следует также иметь в виду, что они могут содержать вредоносный исполняемый код.  
В этом случае в конфигурации следует предусмотреть

* для администратора – дополнительные средства контроля, в частности, список разрешенных (запрещенных) расширений файлов для загрузки в программу;
* блокирование открытия исполняемых файлов из программы (даже если их разрешено загружать и хранить в программе).

Примечание: в общем случае, вредоносный код может содержаться даже в неисполняемых файлах, например, макровирусы в документах Microsoft Office. Однако в этом случае необходимые предупреждение об опасных действиях уже предусмотрены в сторонних приложениях Microsoft Office, поэтому в конфигурации не требуется предпринимать дополнительных мер защиты. Исключение составляет случай открытия через COM – см. [Безопасность программного обеспечения, вызываемого через открытые интерфейсы](https://its.1c.ua/db/content/v8std/src/500/i8100775.htm?_=1580137113).

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) работу с файлами следует организовывать только средствами подсистемы «Работа с файлами». При этом автоматически будут выполнены все требования, перечисленные выше в этом пункте.

6. Безопасность внешних компонент.

6.1. Внешние компоненты, не являющиеся частью конфигурации (не размещенные в макетах конфигурации) потенциально опасны и их не следует загружать из источников, к которым нет доверия, с целью последующей установки и подключения. Пользователи без административных прав не должны иметь возможности загрузки, установки и подключения внешних компонент на сервере прикладного решения. При этом пользователю всегда должен задаваться вопрос и предоставляться выбор, устанавливать ли внешний компонент на клиенте.

Невыполнение этих требований может нарушить работоспособность и безопасность прикладного решения, серверов на которых оно работает и компьютера пользователя.

6.2. Сторонние внешние компоненты следует хранить в специальном справочнике, доступ на запись к которому есть только у администратора и подключать их только по навигационной ссылке на реквизит справочника, в котором хранятся двоичные данные компоненты.

Не следует подключать сторонние внешние компоненты по имени файла или по идентификатору программы, т.к. в этом случае злоумышленник сможет подменить путь к файлу или идентификатор программы и подключить свою вредоносную компоненту.

6.3. Внешние компоненты, входящие в состав конфигурации, должны храниться в макетах типа «Внешняя компонента». Данный тип макета [не локализуется.](https://its.1c.ua/db/content/v8std/src/900/i8100766.htm?_=1580137113)

6.4. При использовании в конфигурации **Библиотеки стандартных подсистем**, следует использовать методы подключения компонент библиотеки и полностью исключить непосредственное использование платформенных механизмов подключения внешних компонент, таких как:

* **ПодключитьВнешнююКомпоненту**;
* **НачатьУстановкуВнешнейКомпоненты**;
* **УстановитьВнешнююКомпоненту**;
* **НачатьПодключениеВнешнейКомпоненты**;
* **ЗагрузитьВнешнююКомпоненту**.

Для подключения компоненты из макета в составе конфигурации на клиенте следует использовать:

ОбщегоНазначенияКлиент.ПодключитьКомпонентуИзМакета

Для подключения компоненты из макета в составе конфигурации на сервере следует использовать:

ОбщегоНазначения.ПодключитьКомпонентуИзМакета

Для подключения компонент из хранилища внешних компонент (специального справочника  с возможностью обновлять компоненты независимо от обновления конфигурации), следует использовать подсистему Внешние компоненты в Библиотеке стандартных подсистем:

ВнешниеКомпонентыКлиент.ПодключитьКомпоненту

7. При загрузке внешнего кода из удаленных источников в конфигурацию, следует:

* использовать только надежные источники, к которым есть доверие;
* выполнять передачу данных только по защищенным каналам связи.

**ЗащищенноеСоединение** = Новый ЗащищенноеСоединениеOpenSSL();  
Соединение = Новый HTTPСоединение(Сервер,,,,,, **ЗащищенноеСоединение**);

При использовании в конфигурации **Библиотеки стандартных подсистем** необходимо использовать функцию **НовоеЗащищенноеСоединение** общего модуля **ОбщегоНазначенияКлиентСервер**:

**ЗащищенноеСоединение** = ОбщегоНазначенияКлиентСервер.НовоеЗащищенноеСоединение();  
Соединение = Новый HTTPСоединение(Сервер,,,,,, **ЗащищенноеСоединение**);

См. также

* [Ограничения на использование Выполнить и Вычислить на сервере](https://its.1c.ua/db/content/v8std/src/500/i8100770.htm?_=1580137113)
* [Облачные технологии](http://v8.1c.ru/overview/Term_000000803.htm) (статья на сайте 1c.ru)
* [Безопасность прикладного программного интерфейса сервера](https://its.1c.ua/db/content/v8std/src/500/i8100678.htm?_=1580137113)

## Ограничения на использование Выполнить и Вычислить на сервере

#std770

Область применения: управляемое приложение, обычное приложение.

1. При разработке решений следует учитывать, что опасно не только непосредственное выполнение кода, написанного в режиме **Предприятие**, но и те места, где методами **Выполнить** или **Вычислить** исполняется код, сконструированный на основе параметров, переданных в серверные функции и процедуры. Ограничение не распространяется на код, выполняемый на клиенте.

Например, код написан следующим образом:

* в клиентской функции в форме создается структура, в которую вставляется строка, написанная разработчиком конфигурации;
* клиентская функция передает эту структуру в серверную функцию формы;
* серверная функция формы вызывает серверную функцию общего модуля;
* в серверной функции исполняется код из строки, вставленной в структуру.

В этом случае не выполняется код, который интерактивно вводит пользователь, но, тем не менее, есть следующая уязвимость:

* злоумышленник создает структуру, в которую вставляет строку с вредоносным кодом;
* злоумышленник вызывает клиентскую функцию формы и таким образом исполняет вредоносный код на сервере.

Еще опаснее, если методы, в которых с помощью **Выполнить** или **Вычислить** исполняется код, принимаемый из параметров, будут располагаться в модулях с установленным признаком ВызовСервера.

2. Для исключения описанных уязвимостей, нужно в серверных процедурах и функциях вызов методов **Выполнить** или **Вычислить** предварять включением безопасного режима:

УстановитьБезопасныйРежим(Истина);  
Выполнить Алгоритм;

В режиме сервиса, включение безопасного режима должно учитывать разделение данных, т.е. приведенный выше пример необходимо доработать следующим образом:

УстановитьБезопасныйРежим(Истина);  
  
Для каждого ИмяРазделителя Из РазделителиКонфигурации() Цикл  
    УстановитьБезопасныйРежимРазделенияДанных(ИмяРазделителя, Истина);  
КонецЦикла;  
  
Выполнить Алгоритм;

При использовании в конфигурации **Библиотеки стандартных подсистем**, следует использовать:

* ОбщегоНазначения.ВыполнитьВБезопасномРежиме() вместо **Выполнить**;
* ОбщегоНазначения.ВычислитьВБезопасномРежиме() вместо **Вычислить**;
* ОбщегоНазначения.ВыполнитьМетодКонфигурации() вместо формирования строки вызова метода модуля и передачи ее в **Выполнить**;
* ОбщегоНазначения.ВыполнитьМетодОбъекта() вместо формирования строки вызова метода объекта и передачи ее в **Выполнить**.

При использовании в конфигурации **Библиотеки стандартных подсистем** версии, меньшей чем 2.4.1, следует использовать:

* РаботаВБезопасномРежиме.ВыполнитьВБезопасномРежиме() вместо **Выполнить**;
* РаботаВБезопасномРежиме.ВычислитьВБезопасномРежиме() вместо **Вычислить**;
* РаботаВБезопасномРежиме.ВыполнитьМетодКонфигурации() вместо формирования строки вызова метода модуля и передачи ее в **Выполнить**;
* РаботаВБезопасномРежиме.ВыполнитьМетодОбъекта() вместо формирования строки вызова метода объекта и передачи ее в **Выполнить**.

3.  Если произвольный код не может быть успешно выполнен в безопасном режиме (например, в нем используется обращение к файлам), то такой код должен предварительно пройти аудит и должен быть размещен в справочнике, к которому есть доступ только у пользователя ответственного за безопасность (например, администратора). Предлагается следующий сценарий работы с внешним кодом:

* внешний код рекомендуется поместить во внешнюю обработку или отчет (в крайнем случае, допустимо передать его, как фрагмент текста);
* передать внешний код администратору с пометкой, что необходимо выполнение кода в небезопасном режиме;
* администратор или сотрудник с соответствующей квалификацией должен выполнить аудит полученного кода;
* если код безопасен и надежен, администратор поместит внешний код в специальный справочник, доступ на запись в который есть только у него;
* вместо **Выполнить** или **Вычислить** следует (или):
  + подключить проверенную обработку из справочника и вызвать необходимый экспортный метод;
  + вызвать фрагмент кода с помощью специальной процедуры или функции, которая централизованно в конфигурации выполняет внешний код.

При использовании в конфигурации **Библиотеки стандартных подсистем**, для этих целей следует воспользоваться подсистемой **Дополнительные отчеты и обработки**.

4. Если конфигурация рассчитана на работу в модели сервиса, и в конфигурации предусмотрен перенос данных в сервис из локальной версии программы, необходимо обеспечить отключение всех пользовательских фрагментов кода или текстов запросов, которые были введены в локальной версии.

При использовании в конфигурации **Библиотеки стандартных подсистем**, также имеется возможность предварительной обработки данных, загружаемых из локальной версии в сервис (см. документацию к подсистеме **Работа в модели сервиса**).

См. также

* [Ограничения на выполнение «внешнего» кода](https://its.1c.ua/db/content/v8std/src/500/i8100669.htm?_=1580137113)
* [Облачные технологии](http://v8.1c.ru/overview/Term_000000803.htm) (статья на сайте 1c.ru)
* [Безопасность прикладного программного интерфейса сервера](https://its.1c.ua/db/content/v8std/src/500/i8100678.htm?_=1580137113)

## Безопасность запуска приложений

#std774

Область применения: управляемое приложение, обычное приложение.

1. При запуске внешней программы из кода требуется составлять строку запуска таким образом, чтобы она собиралась только из проверенных частей.  
Если одна из частей, из которых собирается строка запуска, содержит данные, полученные из базы данных, из поля ввода на форме или прочитаны из хранилища настроек, то перед запуском программы требуется проверить, являются ли запуск безопасным. Безопасными считаются такие строковые данные, которые не содержат в себе следующие символы: "$", "`", "|", "||" ";", "&", "&&".

Данное требование распространяется на все способы запуска программы, в том числе:

* **КомандаСистемы**(<СтрокаКоманды>, <ТекущийКаталог>)
* **ЗапуститьПриложение**(<СтрокаКоманды>, <ТекущийКаталог>, <ДождатьсяЗавершения>, <КодВозврата>) ;
* **НачатьЗапускПриложения**(<ОписаниеОповещения>, <СтрокаКоманды>, <ТекущийКаталог>, <ДождатьсяЗавершения>);
* **ПерейтиПоНавигационнойСсылке**(<НавигационнаяСсылка>);
* Использование **COM** объектов "**Wscript.Shell**" и "**Shell.Application**".

2. При использовании **Библиотеки стандартных подсистем** для запуска внешних программ требуется использовать следующий программный интерфейс:  
2.1. Для того чтобы открыть проводник с фокусировкой на указанном файле, использовать процедуру **ФайловаяСистемаКлиент.ОткрытьПроводник**.  
Например:

// Для Windows  
ФайловаяСистемаКлиент.ОткрытьПроводник("C:\Users");  
ФайловаяСистемаКлиент.ОткрытьПроводник("C:\Program Files\1cv8\common\1cestart.exe");

// Для Linux  
ФайловаяСистемаКлиент.ОткрытьПроводник("/home/");  
ФайловаяСистемаКлиент.ОткрытьПроводник("/opt/1C/v8.3/x86\_64/1cv8c");

2.2. Для того чтобы открыть файл в программе просмотра, ассоциированной с расширением файла, использовать процедуру **ФайловаяСистемаКлиент.ОткрытьФайл**. Она исключает запуск исполняемых файлов (например, \*.exe, \*.bin, \*.apk).  
Например:

ФайловаяСистемаКлиент.ОткрытьФайл(КаталогДокументов() + "test.pdf");  
ФайловаяСистемаКлиент.ОткрытьФайл("D:\test.xlsx");

2.3. Для того чтобы открыть веб-страницу в браузере, запустить программу по протоколу (например, mailto:, skype:, tel: и.т.д) или открыть навигационную ссылку информационной базы следует использовать процедуру **ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку**. При этом в веб-клиенте пользователю будет предложено установить расширение для работы с файлами в тех случаях, когда оно необходимо для выполнения операции.  
Например:

ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("[https://1c.ru](https://1c.ru/)");  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("e1cib/navigationpoint/startpage"); // начальная страница.  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("<mailto:help@1c.ru>");  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("skype:echo123?call");

В то же время, для открытия проводника или файла в программе просмотра не следует формировать ссылку по протоколу file://, для этого следует использовать одну из процедур: **ОткрытьПроводник** (см. п. 2.1) или **ОткрытьФайл** (см. п. 2.2).

2.4. Для того чтобы:

* запускать файлы на исполнение (например, \*.exe, \*bat),
* использовать системные команды (например, ping, tracert или traceroute, обращаться к rac-клиенту),
* выполнять команды на сервере,
* а также получать код возврата и значения потоков вывода (stdout) и ошибок (stderr)

следует использовать **ФайловаяСистемаКлиент.ЗапуститьПрограмму** (в клиентском коде) и **ФайловаяСистема.ЗапуститьПрограмму**(в серверном коде).  
Например:

ФайловаяСистемаКлиент.ЗапуститьПрограмму("calc");

Пример запуска с ожиданием завершения и получения кода возврата:

ПараметрыЗапускаПрограммы = ФайловаяСистема.ПараметрыЗапускаПрограммы();  
ПараметрыЗапускаПрограммы.ДождатьсяЗавершения = Истина;  
ПараметрыЗапускаПрограммы.ПолучитьПотокВывода = Истина;  
ПараметрыЗапускаПрограммы.ПолучитьПотокОшибок = Истина;

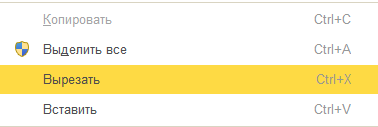
Результат = ФайловаяСистема.ЗапуститьПрограмму(  
 "ping 127.0.0.1 -n 5", ПараметрыЗапускаПрограммы);

КодВозврата = Результат.КодВозврата;  
ПотокВывода = Результат.ПотокВывода;  
ПотокОшибок = Результат.ПотокОшибок;

3. Для выполнения команды, требующей запуск внешней программы в режиме наивысших прав (например, в ОС Windows - с отображением запроса повышения прав UAC), необходимо:

* реализовывать ее на управляемой форме в виде кнопки или пункта меню;
* а на самой кнопке, начинающей выполнение действия, отобразить значок щита (общая картинка **ЗначокЩита** из **Библиотеки стандартных подсистем**).

https://its.1c.ua/db/content/v8std/src/500/i8100774.files/1.png?_=1580137113



См. например, [требования для ОС Windows](https://technet.microsoft.com/ru-ru/library/ff182789.aspx).

## Безопасность программного обеспечения, вызываемого через открытые интерфейсы

#std775

Область применения: управляемое приложение, обычное приложение.

1. При использовании интеграции со сторонними приложениями с помощью открытых интерфейсов (в частности, с помощью COM) требуется отключать исполнение произвольного кода средствами вызываемого приложения.

2. В частности, перед программным открытием документов Microsoft Word и Microsoft Excel через COM следует запрещать исполнение макросов. Иначе это может привести к выполнению вредоносных макросов (вирусов), если таковые присутствуют в документе.

НЕПРАВИЛЬНО открывать документ Microsoft Word по умолчанию:

ОбъектWord = Новый COMОбъект("Word.Application");  
Документ = ОбъектWord.Documents.Open(ИмяФайла);

ПРАВИЛЬНО открывать документ Microsoft Word с отключением макросов:

ОбъектWord = Новый COMОбъект("Word.Application");  
ОбъектWord.WordBasic.DisableAutoMacros(1);  
Документ = ОбъектWord.Documents.Open(ИмяФайла);

ПРАВИЛЬНО открывать документ Microsoft Excel с отключением макросов:

ОбъектExcel = Новый COMОбъект("Excel.Application");  
ОбъектExcel.AutomationSecurity = 3; // msoAutomationSecurityForceDisable = 3  
Документ = ОбъектExcel.Workbooks.Open(ИмяФайла);

3. Если при программном открытии документов Microsoft Office все же необходимо разрешить выполнение автоматически запускающихся макросов, тогда следует реализовать следующую логику работы:

* В конфигурации сделать формы настройки безопасности запуска макросов, отдельно для клиентского и серверного кода со следующими элементами:
  + «Запретить автоматический запуск»;
  + «Разрешить запуск подписанных макросов (рекомендуется)» (выбран по умолчанию);
  + «Разрешить запуск не подписанных макросов (опасно)».
* Форма настройки для клиентского кода должна быть доступна каждому пользователю, настройки должны сохраняться в разрезе пользователей, каждому пользователью должны быть доступны только свои настройки.
* Форма настройки для серверного кода должна быть доступна только администратору, доступ к настройкам должен быть только у администратора.
* При программном открытии документов следует учитывать эти настройки.

3.1. Проверку наличия подписи макросов в документах Microsoft Word можно реализовать так:

ОбъектWord = Новый COMОбъект("Word.Application");  
ОбъектWord.WordBasic.DisableAutoMacros(1); // Отключить автозапуск  
Документ = ОбъектWord.Documents.Open(ФайлДокумента);

Если Документ.**VBASigned** Тогда  
 ОбъектWord.WordBasic.DisableAutoMacros(0); // Включить автозапуск  
 Документ.RunAutoMacro(2); // wdAutoOpen = 2  
Иначе  
 Документ.Close();  
 ВызватьИсключение НСтр("ru = 'Документ не подписан. Открытие отменено.'");  
КонецЕсли;

3.2. Проверку наличия подписи макросов в документах Microsoft Excel можно реализовать так:

ОбъектExcel = Новый COMОбъект("Excel.Application");  
ИсходныйУровеньБезопасности = ОбъектExcel.AutomationSecurity;  
ОбъектExcel.AutomationSecurity = 3; // msoAutomationSecurityForceDisable = 3  
Документ = ОбъектExcel.Workbooks.Open(ФайлДокумента);  
ОбъектExcel.AutomationSecurity = ИсходныйУровеньБезопасности;

Если Документ.**VBASigned** Тогда  
 Документ.Close();  
 Документ = ОбъектExcel.Workbooks.Open(ФайлДокумента);  
Иначе  
 Документ.Close();  
 ВызватьИсключение НСтр("ru = 'Документ не подписан. Открытие отменено.'");  
КонецЕсли;

См. также: [**О параметрах исполнения макросов в Microsoft Excel**](https://msdn.microsoft.com/en-us/vba/excel-vba/articles/application-automationsecurity-property-excel?f=255&MSPPError=-2147217396)

# [Настройка прав доступа к данным](https://its.1c.ua/db/v8std/browse/13/-1/37)

# [Реализация обмена данными](https://its.1c.ua/db/v8std/browse/13/-1/38)

# [Разработка и использование библиотек](https://its.1c.ua/db/v8std/browse/13/-1/39)

# [Требования по локализации](https://its.1c.ua/db/v8std/browse/13/-1/40)

# [Проектирование интерфейсов для 8.3](https://its.1c.ua/db/v8std/browse/13/-1/7)

# [Разработка пользовательских интерфейсов](https://its.1c.ua/db/v8std/browse/13/-1/11)

# [Проектирование интерфейсов для 8.2](https://its.1c.ua/db/v8std/browse/13/-1/15)

# [Разработка пользовательских интерфейсов (обычное приложение)](https://its.1c.ua/db/v8std/browse/13/-1/23)

1. Следует ориентироваться на 1000 строк и более, а также учитывать не только размер таблицы, в которой выполняется поиск, но и сколько раз он выполняется. Например, даже если таблица относительно небольшая в 100 строк, но поиск по ней выполняется 100 раз, ее тоже имеет смысл индексировать. В то же время, нет смысла индексировать таблицу из-за только одной единственной операции поиска. [↑](#footnote-ref-1)
2. Следует ориентироваться на 1000 элементов и более, а также учитывать не только размер массива, но и сколько раз выполняется поиск. Например, если поиск выполняется многократно, в частности, в цикле, то эта рекомендация также действительна для массивов меньшего размера (до 1000 элементов). Особого внимания требуют универсальные механизмы, которые могут применяться на сколь угодно больших объемах данных. [↑](#footnote-ref-2)
3. Следует ориентироваться на 1000 операций конкатенации строк и более (эта величина также может быть еще меньше при увеличении длин строк: чем строки длиннее, тем операции выполняются дольше). Особого внимания требует конкатенация в циклах и в универсальных механизмах, которые могут применяться на сколь угодно больших объемах данных. В то же время, не следует отказываться от конкатенации строк в остальных случаях, так как это заметно снижает читаемость кода. [↑](#footnote-ref-3)
4. Рабочие процессы могут быть также запущены от имени другого пользователя, отличного от того, под которым запускается агент сервера. Подробнее см. руководство администратора клиент-серверного варианта, [описание служебного файла swpuser.ini](http://its.1c.ru/db/v8doc#content:26:1:issogl1_3.20.swpuser.ini) [↑](#footnote-ref-4)
5. Если используется 32-битная версия платформы, и размер результата запроса превосходит размер имеющейся памяти, то данные будут записаны на диск, а затем считаны оттуда в процессе вызовов **Выборка.Следующий()**. [↑](#footnote-ref-5)