Система стандартов и методик разработки конфигураций для платформы

1С:ПРЕДПРИЯТИЕ 8

Версия

январь 2020

Пустая страница

Оглавление

**Элементы оглавления не найдены.**

# [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**.

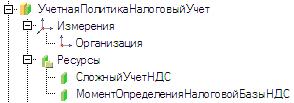
После этого, те или иные объекты конфигурации можно «привязать» к функциональной опции, включив их в ее состав, а в случае необходимости управления доступностью кода – использовать метод **ПолучитьФункциональнуюОпцию**:

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)

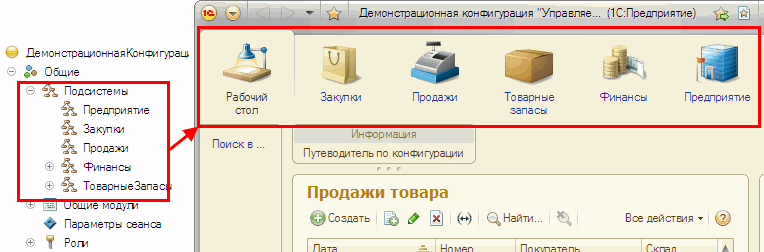
### Использование подсистем

#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.

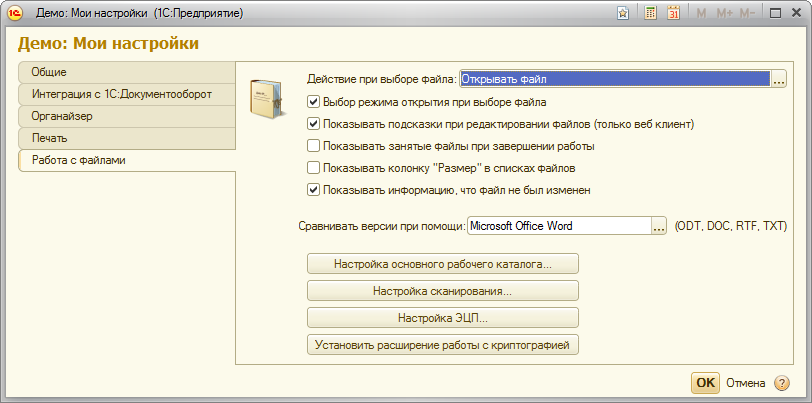
1.2. Для работы с пользовательскими настройками требуется, чтобы для пользователя было доступно право **СохранениеДанныхПользователя**.

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:

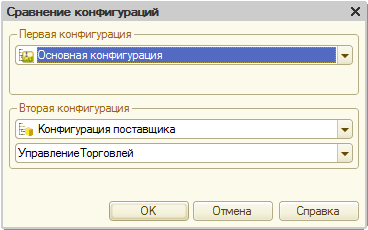
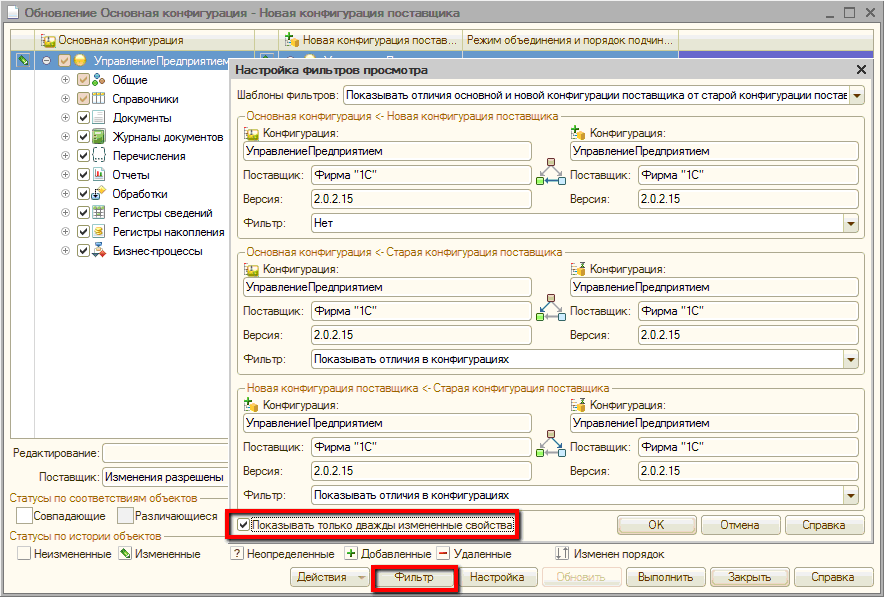
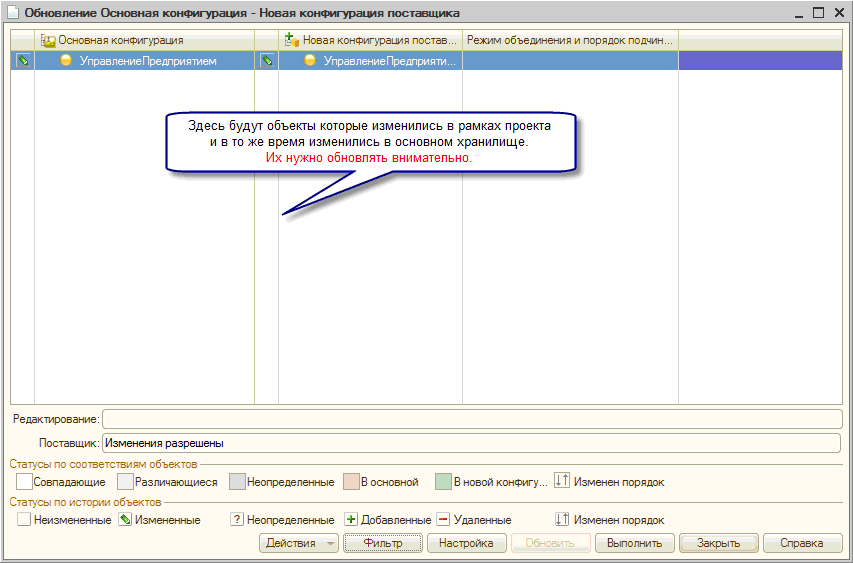
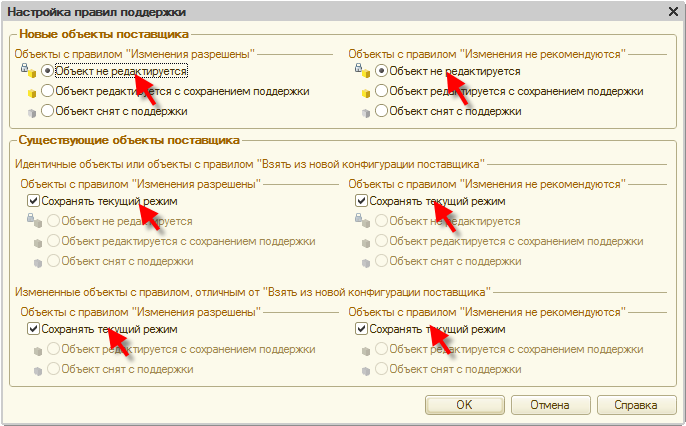
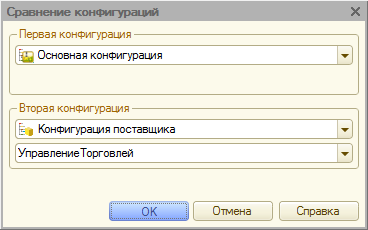
1. В свойствах конфигурации.
2. В процедуре ОбновлениеИнформационнойБазы<ИмяБиблиотеки>.ПриДобавленииПодсистемы (только для конфигураций, основанных на **Библиотеке Стандартных Подсистем**).

#### 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.**

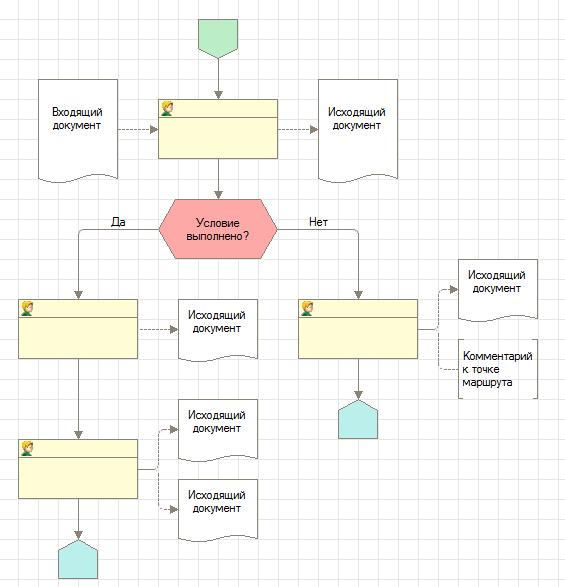
См. также

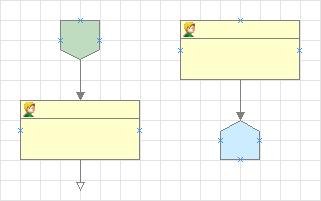
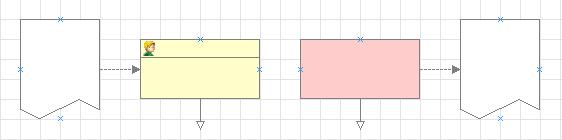
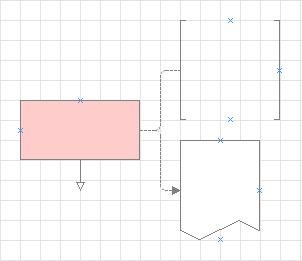
* [Общие требования к конфигурации](https://its.1c.ua/db/content/v8std/src/100/100/i8100467.htm?_=1579516850)
* [Глава "35. Особенности разработки кроссплатформенных прикладных решений"](http://its.1c.ru/db/v837doc#bookmark:dev:TI000001208) документации по платформе 1С:Предприятие

### Оформление карты маршрута бизнес-процесса

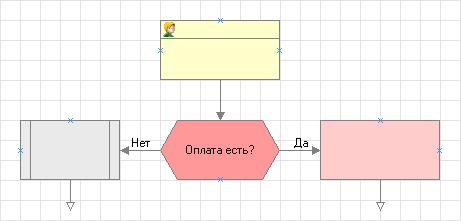
#std480

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

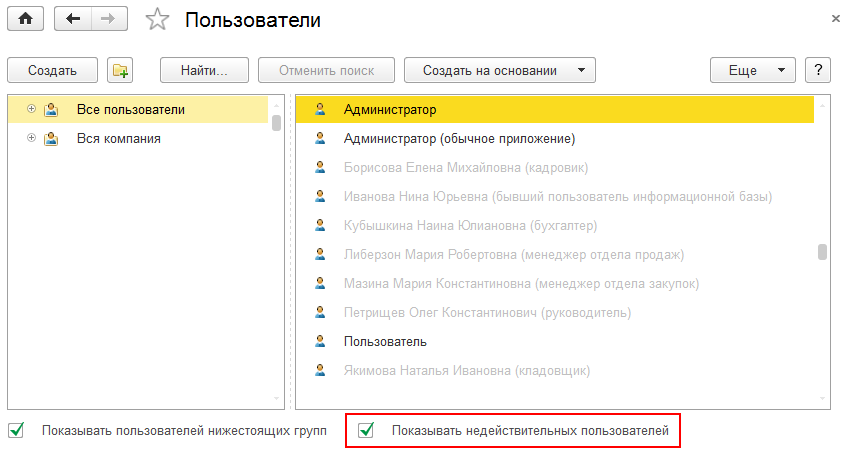
2.1.2. Для тех реквизитов, где это поведение нужно изменить (например, нужно выводить всех пользователей или должно работать другое ограничение) следует явно установить свойства **«Параметры выбора»** и **«Связи параметров выбора»** с необходимыми в конкретном контексте значениями выбора:

* 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

## Оптимизация запросов

## Обработка и модификация данных

## Избыточные блокировки и методы оптимизации

# [Соглашения при написании кода](https://its.1c.ua/db/v8std/browse/13/-1/31)

## Оформление модулей

## Использование конструкций встроенного языка

## Использование прикладных объектов и универсальных коллекций значений

# [Клиент-серверное взаимодействие](https://its.1c.ua/db/v8std/browse/13/-1/35)

# [Общие вопросы безопасности](https://its.1c.ua/db/v8std/browse/13/-1/36)

# [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)