

Система стандартов и методик разработки конфигураций для платформы 1С:Предприятие 8

- [Создание и изменение объектов метаданных](#)
 - [Организация работы конфигурации](#)
 - [Общие требования к конфигурации](#)
 - [Имена объектов метаданных в конфигурациях](#)
 - [Работа в разных часовых поясах](#)
 - [Использование функциональных опций](#)
 - [Использование параметров сеанса](#)
 - [Использование подсистем](#)
 - [Использование общих реквизитов](#)
 - [Использование определяемых типов](#)
 - [Правила создания общих модулей \(Раздел обновлен!\)](#)
 - [Работа с пользовательскими настройками](#)
 - [Версия платформы 1С:Предприятие для разработки \(Новый раздел!\)](#)
 - [Начальные действия при работе конфигурации](#)
 - [Поддержка толстого клиента, управляемое приложение, клиент-сервер \(Раздел обновлен!\)](#)
 - [Технология разветвленной разработки конфигураций](#)
 - [Особенности разработки конфигураций для ОС Linux и macOS \(Раздел обновлен!\)](#)
 - [Оформление карты маршрута бизнес-процесса](#)
 - [Ограничения на переименование объектов метаданных](#)
 - [Требования к установке и обновлению прикладных решений](#)
 - [Несущественные предупреждения проверки конфигурации](#)
 - [Учет версий конфигураций](#)
 - [Общие сведения о выпуске конфигураций](#)
 - [Нумерация редакций и версий](#)
 - [Заполнение свойств конфигурации информацией о выпуске](#)
 - [Организация хранения данных](#)
 - [Общие сведения об организации хранения данных](#)
 - [Уточнение сущности объекта метаданных](#)
 - [Имя, синоним, комментарий](#)
 - [Подсказка и проверка заполнения](#)
 - [Использование кодов \(номеров\) объектов конфигурации](#)
 - [Использование реквизитов строкового типа](#)
 - [Ограничения на использование реквизитов составного типа \(Раздел обновлен!\)](#)
 - [Требования к проведению документов](#)
 - [Использование активности, движений](#)
 - [Самодостаточность регистров](#)
 - [Реквизит "Комментарий" у документов](#)
 - [Удаление устаревших объектов метаданных из конфигурации](#)
 - [Использование констант](#)
 - [Работа с неактуальными \(недействительными\) объектами](#)
 - [Использование предопределенных элементов](#)
 - [Обработчики событий объектов](#)
 - [Обработчик события ПередЗаписью](#)
 - [Обработчик события ПриЗаписи](#)
 - [Обработчик события ПередУдалением](#)
 - [Обработчик события ПриКопировании](#)
 - [Обработчик события ОбработкаПроверкиЗаполнения](#)
 - [Обработчик события ОбработкаЗаполнения](#)
 - [Обработчик событий ОбработкаПолученияПредставления и ОбработкаПолученияПолейПредставления](#)
 - [Использование признака ОбменДанными.Загрузка в обработчиках событий объекта](#)
 - [Регламентные задания](#)
 - [Общие требования к регламентным заданиям](#)
 - [Настройка расписания регламентных заданий](#)
 - [Запуск регламентных заданий](#)
 - [Ограничения на регламентные задания при работе в режиме сервиса](#)
 - [Реализация обработки данных](#)
 - [Работа с запросами](#)
 - [Оформление текстов запросов](#)
 - [Многократное выполнение однотипных запросов](#)
 - [Проверка на пустой результат выполнения запроса](#)
 - [Ограничение на использование конструкции "ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ" в запросах](#)
 - [Использование ключевых слов "ОБЪЕДИНИТЬ" и "ОБЪЕДИНИТЬ ВСЕ" в запросах](#)
 - [Упорядочивание результатов запроса](#)
 - [Округление результатов арифметических операций в запросах](#)
 - [Особенности использования в запросах оператора ПОДОБНО](#)
 - [Псевдонимы источников данных в запросах](#)
 - [Оптимизация запросов](#)
 - [Общие требования по разработке оптимальных запросов](#)
 - [Несоответствие индексов и условий запроса](#)
 - [Разыменование ссылочных полей составного типа в языке запросов](#)
 - [Ограничения на соединения с вложенными запросами и виртуальными таблицами](#)
 - [Ограничения на использование вложенных запросов в условии соединения](#)
 - [Обращения к виртуальным таблицам](#)
 - [Эффективные условия запросов](#)
 - [Разрешение итогов для периодических регистров сведений](#)
 - [Эффективное обращение к виртуальной таблице «Остатки»](#)
 - [Использование временных таблиц](#)
 - [Обработка и модификация данных](#)
 - [Транзакции: правила использования \(Новый раздел!\)](#)
 - [Использование управляемого режима блокировки](#)
 - [Блокировка данных объекта для редактирования из кода](#)
 - [Ответственный чтение данных \(Раздел обновлен!\)](#)
 - [Чтение отдельных реквизитов объекта из базы данных](#)
 - [Запись событий в историю работы пользователя](#)
 - [Избыточные блокировки и методы оптимизации](#)
 - [Общие сведения об избыточных блокировках](#)
 - [Сдвиг границы последовательности документов](#)
 - [Режим разделения итогов для регистров бухгалтерии](#)
 - [Режим разделения итогов для регистров накопления](#)
 - [Блокирующее чтение остатков в начале транзакции](#)
 - [Соглашения при написании кода](#)
 - [Оформление модулей](#)
 - [Тексты модулей \(Раздел обновлен!\)](#)
 - [Структура модуля](#)
 - [Имена процедур и функций](#)
 - [Описание процедур и функций \(Раздел обновлен!\)](#)
 - [Параметры процедур и функций](#)
 - [Особенности использования структур в качестве параметров процедур и функций](#)
 - [Правила образования имен переменных](#)

- Работа с параметром «Отказ» в обработчиках событий
- Использование конструкций встроенного языка
 - Общие требования к построению конструкций встроенного языка
 - Перенос выражений
 - Использование дублирующего кода
 - Использование директив компиляции и инструкций препроцессора
 - Определение типа значения переменной
 - Получение метаданных объектов
 - Обработчики событий модуля формы, подключаемые из кода
 - Использование глобальных переменных в программных модулях
 - Предварительная инициализация локальных переменных
 - Использование Журнала регистрации
 - Перехват исключений в коде (Раздел обновлен!)
 - Ограничение на использование оператора Перейти
- Использование прикладных объектов и универсальных коллекций значений
 - Поиск в коллекциях значений
 - Использование объекта РегистрСведенийМенеджерЗаписи
 - Копирование строк между таблицами значений (табличными частями и т.п.) произвольной структуры
 - Порядок записи движений документов
 - Получение представлений для ссылочных значений в табличном документе
 - Программное создание прикладных объектов
 - Использование модуля объекта, модуля менеджера объекта и общих модулей
 - Ограничения на использование экспортных процедур и функций
 - Установка параметров выбора и связей параметров выбора для объектов металанных
 - Использование РеквизитФормыВЗначение и ДанныеФормыВЗначение
 - Применение параметров отчета в СКД
 - Использование объектов типа Структура
 - Особенности сортировки в таблице значений
 - Массовая конкатенация строк
- Клиент-серверное взаимодействие
 - Использование модулей с повторным использованием возвращаемых значений
 - Использование значений, влияющих на поведение клиентского приложения
 - Получение предопределенных значений на клиенте
 - Минимизация количества серверных вызовов (Раздел обновлен!)
 - Минимизация количества выполняемого на клиенте
 - Доступ к файловой системе из кода конфигурации (Раздел обновлен!)
 - Оптимизация использования оперативной памяти
 - Таймауты при работе с внешними ресурсами
- Общие вопросы безопасности
 - Безопасность прикладного программного интерфейса сервера
 - Ограничение на установку признака «Вызов сервера» у общих модулей
 - Безопасное хранение паролей (Раздел обновлен!)
 - Ограничение на выполнение внешнего кода
 - Ограничения на использование Выполнить и Вычислить на сервере
 - Безопасность запуска приложений (Раздел обновлен!)
 - Безопасность программного обеспечения, вызываемого через открытые интерфейсы
- Настройка прав доступа к данным
 - Настройка ролей и прав доступа (Раздел обновлен!)
 - Стандартные роли
 - Установка прав для новых объектов и полей объектов
 - Проверка прав доступа
 - Использование привилегированного режима
 - Ограничение на использование ключевого слова "РАЗРЕШЕННЫЕ" в запросах
 - Влияние изменения значений параметров сеанса и функциональных опций на производительность механизма ограничения доступа к данным
- Реализация обмена данными
 - Настройка обмена данными для классификаторов между различными информационными базами
 - Разработка планов обмена с отборами
 - Интеграция прикладных решений через формат EnterpriseData
- Разработка и использование библиотек
 - Разработка конфигураций с повторным использованием общего кода и объектов металанных
 - Имена объектов метаданных в иерархии библиотек
 - Переопределяемые и поставляемые объекты библиотеки
 - Отнесение библиотечных объектов к подсистемам
 - Переопределение общих модулей в условиях иерархии библиотек
 - Размещение сведений о настройках подсистемы
 - Обеспечение совместимости библиотек
 - Разработка ролей в библиотеках
 - Обработчики обновления информационной базы (БСП).
- Требования по локализации
 - Общие требования по локализации конфигурации (Раздел обновлен!)
 - Поставка международной версии конфигурации (Раздел обновлен!)
 - Интерфейсные тексты в коде: требования по локализации (Раздел обновлен!)
 - Запросы, динамические списки и отчеты на СКД; требования по локализации
 - Форматирование даты, числа, Булево: требования по локализации (Раздел обновлен!)
 - Строковые константные выражения в коде: требования по локализации
 - Элементы форм: требования по локализации (Раздел обновлен!)
 - Регламентные задания: требования по локализации
 - Макеты: требования по локализации (Раздел обновлен!)
 - Денежные поля: требования по локализации
 - Автогенерированные данные в информационной базе: требования по локализации (Новый раздел!)
- Проектирование интерфейсов для 8.3
 - Размеры экрана
 - Оформление групп разделов с настройками и справочниками
 - Командный интерфейс
 - Общие принципы построения командного интерфейса
 - Панель разделов
 - Навигация внутри раздела
 - Как вместить большое количество команд
 - Компоновка форм
 - Формы документов
 - Командная панель документа
 - Табличные части. Оформление списка
 - Итоги в документах
 - Поля "Ответственный" и "Комментарий"
 - Элементы интерфейса
 - Тумблер
 - Подсказки на форме
 - Шрифты
- Разработка пользовательских интерфейсов
 - Пользовательские представления объектов
 - Использование сочетаний клавиш, список зарезервированных сочетаний
 - Длительные операции на сервере (Раздел обновлен!)
 - Длительные операции на клиенте
 - Формирование печатных форм

- [Реализация работы формы](#)
 - [Открытие форм](#)
 - [Открытие параметризованных форм](#)
 - [Правила создания модулей форм](#)
 - [Блокирующее или независимое открытие форм объектов](#)
 - [Ограничения на использование модальных окон и синхронных вызовов](#)
 - [Запрет редактирования полей таблицы по условию](#)
 - [Особенности табличного документа в веб-клиенте](#)
 - [Обращение из кода к автоматически формируемым элементам управления формы](#)
 - [Обращение из кода к пользовательским элементам управления формы](#)
 - [Команды по модификации объектов](#)
 - [Контекстная и внеоконтекстная передача управления на сервер](#)
 - [Использование объекта `ДанныеФормыКоллекция`](#)
 - [Условное оформление в формах](#)
 - [Ограничение использования поля HTML_документа](#)
 - [Использование режима вертикальной прокрутки форм](#)
 - [История выбора при вводе](#)
- [Реализация форм списков](#)
 - [Ограничения при использовании динамических списков](#)
 - [Особенности реализации команд для форм списков](#)
 - [Организация работы со списками данных с помощью общих команд](#)
 - [Обновление списков при интерактивных действиях пользователя](#)
 - [Реквизит Ссылка в динамических списках](#)
 - [Запросы в динамических списках](#)
 - [Поле "Дата" в списках](#)
 - [Программное переопределение текстов запросов динамических списков](#)
- [Организация диалога с пользователем](#)
 - [Информирование пользователя \(Раздел обновлен!\)](#)
 - [Ограничение на использование метода Сообщить](#)
 - [Предложение об установке внешних компонент и расширений платформы](#)
- [Проектирование интерфейсов для 8.2](#)
 - [Общие рекомендации](#)
 - [Командный интерфейс](#)
 - [Панель разделов](#)
 - Панель разделов
 - Названия разделов
 - Картинки разделов
 - Подсказки для интерфейсных подсистем
 - [Панель навигации основного окна](#)
 - Панель навигации основного окна
 - Порядок и названия команд в ПН
 - Группа команд «Важное» в ПН
 - Группировка команд в ПН
 - Группа «См. также» в ПН
 - Команды, размещаемые в ПН
 - [Панель действий](#)
 - [Отчеты](#)
 - Содержание отчета
 - Варианты отчетов
 - Поля периодов
 - Пользовательские настройки
 - Заголовок отчета
 - Отчеты вида "таблица", "список"
 - Отчеты вида "диаграмма"
 - [Размещение большого количества команд в основном окне приложения](#)
 - [Рабочий стол](#)
 - [Рабочее место](#)
 - [Оформление форм списков](#)
 - Формы списков
 - Списки с одной колонкой
 - Размеры списков
 - Быстрые отборы в списках
 - Многоэтажные списки
 - Команды, размещаемые во "Всех действиях"
 - Заголовки списков
 - Колонки с флагами
 - Группировки в списках
 - Команда "Создать" в журналах документов
 - Пояснение невозможности заполнения ячеек в табличных частях
 - Акцентирование внимания на просроченных или критичных состояниях
 - Групповые обработки в списках
 - [Сообщения пользователю](#)
 - [Окно старта](#)
 - [Формы](#)
 - Формы
 - Компоновка форм
 - Командная панель формы
 - Порядок полей
 - Размеры
 - Группы элементов формы
 - Поля "Ответственный" и "Комментарий"
 - Группа полей "Наименование", "Код", "Полное наименование", "Входит в группу"
 - Панель навигации вспомогательного окна
 - Список, открываемый из панели навигации формы объекта
 - Блокирующая или независимая форма
 - Формы выбора
 - Формы пошаговых помощников (мастеров)
 - Взаимосвязанные поля
 - Командные панели табличных частей
 - [Оформление элементов](#)
 - Оформление элементов
 - Переход к форме с дополнительными реквизитами
 - Выбор: Гиперссылка или кнопка
 - Картинки (иконки) в названии команд
 - Частотные кнопки
 - Итоги в документах
 - Итоги в журналах документов
 - Флажки
 - Команда «Подобрать»
 - Команда «Отмена»
 - Единицы измерения
 - Значения по умолчанию
 - Гиперссылка на счет-фактуру
 - Поле, влияющее на состав остальных полей в форме

- [Невыбранная картинка](#)
- [Реквизиты](#)
- [Подменю](#)
- [Требования к изображениям \(иконкам\)](#)
- [Правила создания иконок командных панелей](#)
- [Тексты](#)
 - [Названия печатных форм учетных документов и команд по их выводу на печать](#)
 - [Тексты](#)
 - [Шрифт и цвет](#)
- [Горячие клавиши](#)
- [Элементы стиля](#)
- [Разработка пользовательских интерфейсов \(обычное приложение\)](#)
 - [Общие правила построения интерфейсов](#)
 - [Общие интерфейсы](#)
 - [Интерфейс "Полный"](#)
 - [Стили](#)
 - [Реализация работы формы](#)
 - [Имя элемента управления](#)
 - [Изменение размера колонки табличного поля](#)
 - [Ограничения по использованию одинаковых текстов на элементах управления в форме](#)
 - [Размеры формы](#)
 - [Подсказки](#)
 - [Использование флагов "Автвыбор незаполненного" и "Автоотметка незаполненного"](#)
 - [Использование гиперссылок в диалогах форм](#)
 - [Привязки](#)
 - [Отступы](#)
 - [Использование закладок](#)
 - [Порядок обхода элементов диалога](#)
 - [Размещение кнопки вызова справки в формах](#)
 - [Отображение единственного табличного поля в форме](#)
 - [Разделители](#)
 - [Кнопки](#)
 - [Картинки](#)
 - [Программное управление видимостью страниц](#)
 - [Программное управление формой](#)
 - [Ограничение выполнения действий, доступных только при определенных условиях](#)
 - [Поведение специализированных форм](#)
 - [Обращение к данным информационной базы в обработчиках часто вызываемых событий](#)
 - [Обращение к свойству "ТекущаяСтрока" табличного поля](#)
 - [Использование пояснений в полях ввода и выбора](#)
 - [Особенности размещения в командных панелях пунктов меню, не предназначенные для решения основных задач](#)
 - [Организация диалога с пользователем](#)
 - [Вопрос при закрытии программы](#)

Общие требования к конфигурации

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

#std467

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

Область применения (уточнение): управляемое приложение, обычное приложение.

взаимодействие с пользователем должно быть организовано асинхронно.
Дополнительные материалы:

- [Список поддерживаемых операционных систем и СУБД](#)
- Особенности работы с различными СУБД см. в [приложении 8](#) документации по платформе 1С:Предприятие 8.3
- Особенности режима низкой скорости соединения и работы веб-клиента см. в [приложении 7](#) (там же)
- [Особенности разработки конфигураций для ОС Linux](#)

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;](#)
- [Ограничение на использование модальных окон и синхронных вызовов;](#)
- [Ограничение на установку признака «Вызов сервера»;](#)
- [Несущественные предупреждения проверки конфигурации.](#)

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](#).
- 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.

Иключение составляют:

- [общие реквизиты](#) (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"](#) (англ. "Obsolete"), которые допустимо размещать в конце соответствующей ветки метаданных;

Имена объектов метаданных в конфигурациях

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

#std550

См. также: [общие правила наименования метаданных](#)

п/п	Объекты метаданных	Правила наименования	Область применения (уточнение)
1.	Подсистемы	Согласно общим правилам наименования метаданных. Например: Финансы , Маркетинг , НастройкаАдминистрирование . См. также: Использование подсистем	Управляемое приложение Обычное приложение
2.	Общие модули	См. Правила создания общих модулей	
3.	Параметры сеанса	Согласно общим правилам наименования метаданных. Например: ТекущийПользователь , ОбменДаннымиВключен , РаботаСВнешнимиРесурсамиЗаблокирована . См. также: Использование параметров сеанса	
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 , ИнтерактивноеОткрытиеВнешнихОтчетов и Обработок . См. также: Стандартные роли	
5.	Общие реквизиты	Согласно общим правилам наименования метаданных.	Управляемое приложение Обычное приложение
6.	Планы обмена	Имена планов обмена рекомендуется называть по следующим принципам: <ul style="list-style-type: none"> 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, ОбменУправлениеГорловлейРозничнаяТорговля. 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 ЗапросРедактированияРеквизитовОбъектовПередЗаписьюОбъекта 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 выключающейся) 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	
11.	Параметры функциональных опций	Имена параметров функциональных опций рекомендуется задавать от описания параметра. При этом не обязательно, 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: <ul style="list-style-type: none"> • РегистрСведений.НазначениеДополнительныхОбработок.Измерение.ТипОбъекта, и • РегистрСведений.НастройкаВерсионированияОбъектов.Измерение.ТипОбъекта См. также: Использование функциональных опций	
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.	Управляемое приложение Обычное приложение

		См. также: Использование определяемых типов	
13.	Хранилища настроек	Согласно общим правилам наименования метаданных. Например: ХранилищеVariantовОтчетов .	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: <ul style="list-style-type: none"> 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, ЗадачиПоBusinessПроцессу 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 интерфейсы	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 , КомпонентаТWAIN , ОписаниеИзмененийСистемы .	
19.	Общие картинки	Согласно общим правилам наименования имен метаданных. Например: Найти – универсальная картинка для команд поиска, для использования в различных подсистемах конфигурации. ЗакрепитьVariantОтчета – 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: КоллекцияVariantыВажностиЗадачи . При этом следует избегать в имени общих картинок слов, от удаления которых смысл не меняется, например: «Картина...», «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...», «XDTO...». 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: GetCurrentRate	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: Стили	Управляемое приложение Обычное приложение
24.	Стили	Согласно общим правилам наименования метаданных. См. также: Стили (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: <ul style="list-style-type: none"> 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...».	

		<p>При выборе имени документа следует различать два случая:</p> <ol style="list-style-type: none"> 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. <p>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.</p> <ol style="list-style-type: none"> 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. <p>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.</p>	
29.	Журналы документов	<p>Имена журналов документов рекомендуется задавать во множественном числе и образовывать от описания списка объектов, которые содержатся в журнале. Например: СкладскиеДокументы, КорректировкиНДС.</p> <p>При этом следует избегать в именах слов, от удаления которых смысл не меняется, например: «Документы...».</p>	
30.	Перечисления	<p>Имена перечислений в конфигурации рекомендуется задавать во множественном числе.</p> <p>Неправильно ДействиеСДокументамиПодДвойномуЩелчу ВажностьЗадачи SMTPAутентификация</p> <p>Правильно: ДействияСДокументамиПодДвойномуЩелчу ВариантыВажностиЗадачи ВидыSMTPAутентификации</p> <p>Любое исключение из этого правила должно быть обоснованным. Например: ПолФизическогоЛица</p>	
31.	Отчеты	<p>1. Имена отчетов и вариантов отчетов рекомендуется образовывать от имени существительного. Например: ДинамикаИзмененийФайлов, СписокЗависящихЗадач, СправкаОбИсполнительскойДисциплине.</p> <p>2. Рекомендуется предусматривать вывод заголовка, если отчет или вариант отчета предназначен для печати.</p> <p style="text-align: right;"><i>Методическая рекомендация (полезный совет)</i></p> <p>Для отчетов с макетом заголовок должен располагаться в самом макете. Для вариантов отчетов без макета достаточно установить свойство "Заголовок" на закладке "Дополнительные настройки".</p> <p>3. При выборе представления варианта отчета следует придерживаться следующих рекомендаций:</p> <ul style="list-style-type: none"> Представление должно лаконично и однозначно описывать суть данных, которые выводятся в этом варианте отчета. <p>Например, для отчета «Валовая прибыль», неправильно: «Основной», «По заказам» (подобные названия могут встречаться у вариантов других отчетов)</p> <p>Правильно: «Валовая прибыль по контрагентам», «Валовая прибыль по заказам»</p> <ul style="list-style-type: none"> 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. <p>4. При этом следует избегать в именах отчетов и вариантов отчетов слов, от удаления которых смысл не меняется, например: «Отчет...».</p>	Управляемое приложение Обычное приложение
32.	Обработки	<p>Имена обработок рекомендуется образовывать от имени существительного. Например: КонтрольЖурналаРегистрации, РегламентныеИФоновыеЗадания, УправлениеИтогамиИАгрегатами.</p> <p>При этом следует избегать в именах обработок слов, от удаления которых смысл не меняется, например: «Обработка...».</p> <p>Если форма обработки предполагает 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.</p>	
33.	Планы видов характеристик	<p>Имена планов видов характеристики рекомендуется задавать во множественном числе и образовывать от описания списка объектов, которые перечисляются в плане видов характеристик.</p> <p>Например: ВидыДоступа, ВопросыДляАнкетирования, ДополнительныеРеквизитыИСведения</p>	Управляемое приложение Обычное приложение
34.	Планы счетов	<p>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.</p> <p>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.</p> <p>Для дополнительных уточнений можно использовать свойство Пояснение, значение которого выводится в подсказке к команде открытия плана счетов.</p> <p>Например:</p> <ul style="list-style-type: none"> 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	<p>Имена планов видов расчета рекомендуется задавать во множественном числе и образовывать от описания списка</p>	Управляемое приложение

		объектов, которые перечисляются в плане видов расчета. Например: ОсновныеНачисления , УправленическиеНачисления , Удержания .	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: ДоговорыУправленическойУчетнойСистемы , 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](#)

Работа в разных часовых поясах

Область применения: управляемое приложение, обычное приложение.

#std643

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 пользователя 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)
    Текущие данные = Элементы.Список.Текущие данные;
    Если Текущие данные = Неопределено Тогда
        ТекДата = ТекущаяДата(); // вызов ТекущаяДата() на клиенте
    Иначе
        ТекДата = Текущие данные.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)
    Текущие данные = Элементы.Список.Текущие данные;
    Если Текущие данные = Неопределено Тогда

```

```
    Текдата = Неопределен; // 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();
ДатаСледующегоПовещения = РассчитатьДату() + ТекущаяДатаСеанса();
```

Правильно

использовать ранее рассчитанные дату и время:

```
ДатаПоследнегоПовещения = ТекущаяДатаСеанса();
ДатаСледующегоПовещения = РассчитатьДату() + ДатаПоследнегоПовещения;
```

Использование функциональных опций

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

#std470

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**:

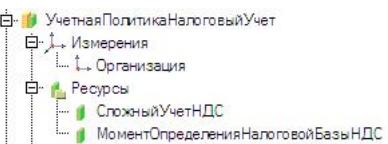
ИспользуетсяМеханизмВерсионирования = ПолучитьФункциональнуюОпцию("ИспользоватьВерсионированиеОбъектов");

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](#).

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 параметризовывать 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 понято 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 встроенным 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](#)

Использование параметров сеанса

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

#std413

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 - Стока - имя параметра сеанса, который требуется установить (принициализировать).
// УстановленныеПараметры - Массив - массив, в который добавляются имена
// установленных (принициализированных) 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](#)

Использование подсистем

Область применения: управляемое приложение, обычное приложение.

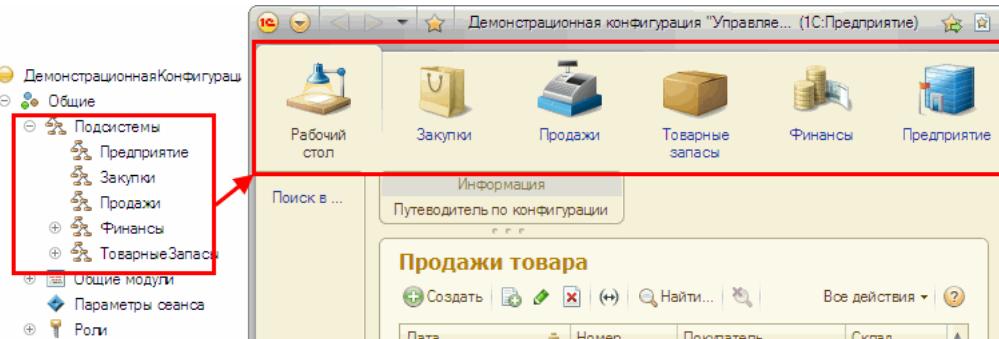
#std543

Методическая рекомендация (полезный совет)

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 флагок **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 флагок **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

Использование общих реквизитов

Область применения: управляемое приложение, обычное приложение.

#std677

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](#), которые связаны с ними.

См. также

- [Использование общих реквизитов без разделения данных](#) (статья на ИТС)

Использование определяемых типов

Область применения: управляемое приложение, обычное приложение.

#std704

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»](#)

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 одноковую длину, точность данных во всех местах использования, упрощает доработку в случае изменения требований.

Например:

- **НомерСчетаФактуры** - Стока, длина 30. Регламентирует формат номера счета-фактуры в различных документах: **ПоступлениеТоваровИУслуг**, **ЗаписьКнигиПокупок**, **ВозвратТоваровОтКлиента** и др.
- **АдресДоставки** - Стока, 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](#), 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**, предусмотренным 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 составной тип, состоящего из единственного типа **Контрагенты**, на «всякий случай», для «механического» упрощения возможного изменения конфигурации в дальнейшем, поскольку это размыает прикладной смысл сущности.

Правила создания общих модулей

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

#std469

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](#) (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** снят),
- 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](#).

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**.

клиентский программный интерфейс сервера приложения.

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](#) 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](#)

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](#)

избежать дублирования кода, 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](#). 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](#)

Особым случаем смешанных клиент-серверных модулей являются [модули форм](#) 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](#)

локализуемых конфигурациях, на базе которых выпускаются национальные прикладные решения для различных стран или регионов, модули, реализующие национальную специфику, именуются с постфиксами "**Локализация**" (англ. "**Localization**") и "**КлиентЛокализация**" (англ. "**ClientLocalization**").

Например: ЭлектроннаяПодписьЛокализация, **ElectronicSignatureLocalization**.

См. также

- Тексты модулей
- Структура модуля

Работа с пользовательскими настройками

Область применения: управляемое приложение, обычное приложение.

#std557

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**.

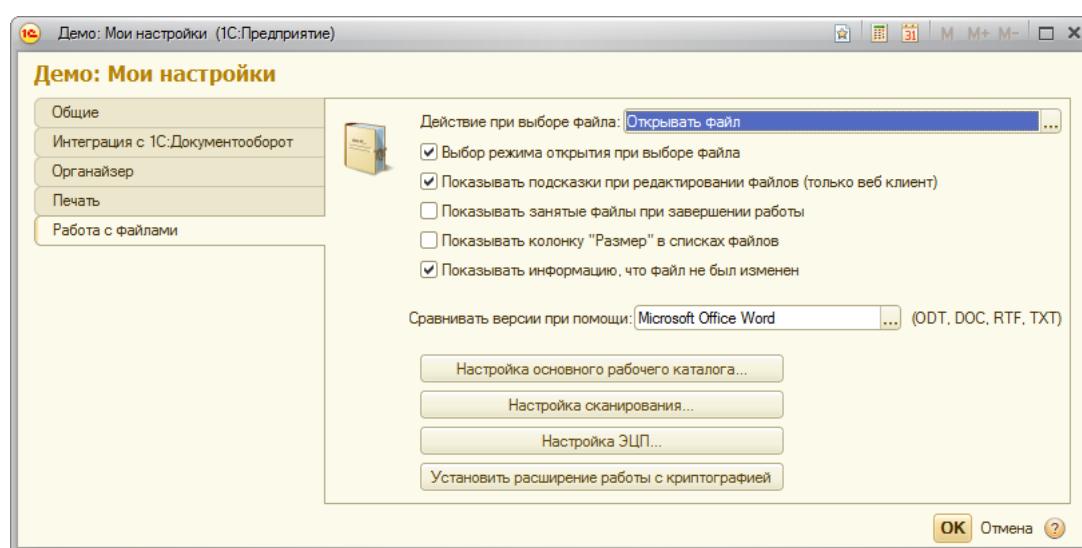
См. также: [Стандартные роли](#)

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, флагок «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](#)

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.

Версия платформы 1С:Предприятие для разработки

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

#std785

1. Минимальная версия платформы, на которой обеспечивается работа прикладного решения, устанавливается:

- не больше номера версии платформы 1С:Предприятие, на которой основан последний сертифицированный релиз платформы – в прикладных решениях для государственных учреждений, оборонных предприятий и корпоративных клиентов, а также в любых библиотеках (которые потенциально могут входить в подобные прикладные решения). См. [зашieldенный программный комплекс "1С:Предприятие 8.3z"](#).
- или на усмотрение ответственных за конфигурацию – в остальных прикладных решениях.

При этом в качестве минимальной версии выбирается та версия платформы, на основе которой собран последний сертифицированный релиз 1С:Предприятие 8.3z. Номер последнего сертифицированного релиза, а также номер версии платформы, который принимается за минимальную версию публикуется на [странице 1С:Предприятие 8.3z на сайте online.1c.ru](#).

Например, если прикладное решение для государственных учреждений выпускается на версии платформы 1С:Предприятие 8.3.15 в режиме совместимости с 8.3.14 и последний сертифицированный релиз 8.3.14.2032, который основан на версии 8.3.14.1976, то за минимальную версию платформы выбирается версия 8.3.14.1976.

1.1. При невозможности обойти ошибку на уровне прикладных решений, минимальная версия платформы может быть увеличена (при этом для прикладных решений для государственных учреждений новый релиз платформы должен быть сертифицирован в короткие сроки).

1.2. В исправительных релизах библиотек не допускается увеличивать минимальную версию платформы, т.к. это решение должно приниматься с учетом всех возможных последствий для каждого конкретного прикладного решения.

2. Рекомендуемая версия платформы устанавливается:

- в прикладных конфигурациях – на усмотрение ответственного за конфигурацию; стоит ориентироваться на последний опубликованный релиз платформы, на котором ведется разработка и тестирование;
- не может быть увеличена в исправительных релизах библиотек, т.к. это решение должно приниматься с учетом всех последствий для каждого конкретного прикладного решения.

Например, прикладное решение выпускается на минимальной версии платформы 1С:Предприятие 8.3.14, а рекомендуемой версией указан последний опубликованный релиз 8.3.15, т.к. в нем улучшен ряд функций, которые важны для пользователей этого прикладного решения.

В прикладных решениях для государственных учреждений, оборонных предприятий и корпоративных клиентов номер рекомендуемой версии платформы может быть больше номера версии, на которой основан последний сертифицированный релиз. В таком случае пользователи могут продолжить работу на текущем сертифицированном релизе платформы и одновременно будут проинформированы об этой рекомендации.

3. Запуск на платформе ниже минимальной версии блокируется с помощью 1С:Библиотеки стандартных подсистем (БСП). Минимальная и рекомендуемая версии платформы задаются в процедуре **ПриОпределенииОбщихПараметровБазовойФункциональности** общего модуля **ОбщегоНазначенияПереопределяемый**.

Минимальная и рекомендуемая версии платформы также указываются в файле `readme.txt`, входящем в состав дистрибутива.

Например:

Минимальная версия платформы 1С:Предприятие для использования конфигурации: 8.3.14.1976.

Рекомендуемая версия платформы: 8.3.15.1700 (или больше).

Начальные действия при работе конфигурации

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

#std556

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](#) 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:

```
ПрофильтОбъект = Справочники.ПрофилиГруппДоступа.СоздатьЭлемент();
ПрофильтОбъект.Наименование = НСтр("ru = 'Бухгалтер'");
ПрофильтОбъект.Предустановленный = Истина;
ПрофильтОбъект.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
        ПрофильтОбъект = Справочники.ПрофилиГруппДоступа.СоздатьЭлемент();
        ПрофильтОбъект.Наименование = НаименованиеПрофиля;
        ПрофильтОбъект.Предустановленный = Истина;
        ПрофильтОбъект.Записать();
    КонецЕсли;
```

Поддержка толстого клиента, управляемое приложение, клиент-сервер

Область применения: управляемое приложение, обычное приложение.

#std680

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](#).

2. При этом разработка конфигураций, рассчитанных на режим управления 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, 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](#). 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 флаг **Вызов сервера**. Подробнее см. [Ограничение на установку признака «Вызов сервера» у общих модулей](#).

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](#).

Технология разветвленной разработки конфигураций

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

#std709

Методическая рекомендация (полезный совет)

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) рекомендуется 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 было 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 основное 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 создано автоматически. Если СППР не используется, хранилище технического проекта нужно будет создавать вручную, в соответствии 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 точно – 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](#)

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 Управление Предприятием и Управление Торговлей), 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 Обновление Информационной Базы <ИмяБиблиотеки>.При Добавлении Подсистемы (только для конфигураций, основанных на **Библиотеке Стандартных Подсистем**).

Приложение 1. Порядок создания хранилища технического проекта

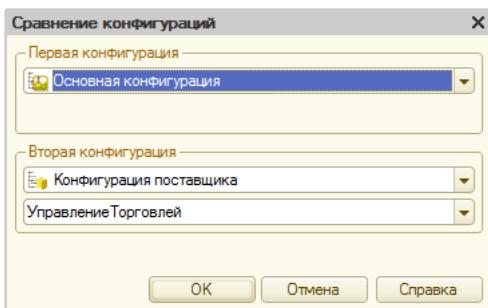
1. Обновить из хранилища конфигурацию информационной базы, подключенную к основному хранилищу
2. Создать файл поставки конфигурации основного хранилища (*.cf)
3. В информационную базу, которая будет использоваться для работы над техническим проектом, загрузить конфигурацию из файла поставки. После загрузки 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 Только Просмотр (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 Только Просмотр.

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.

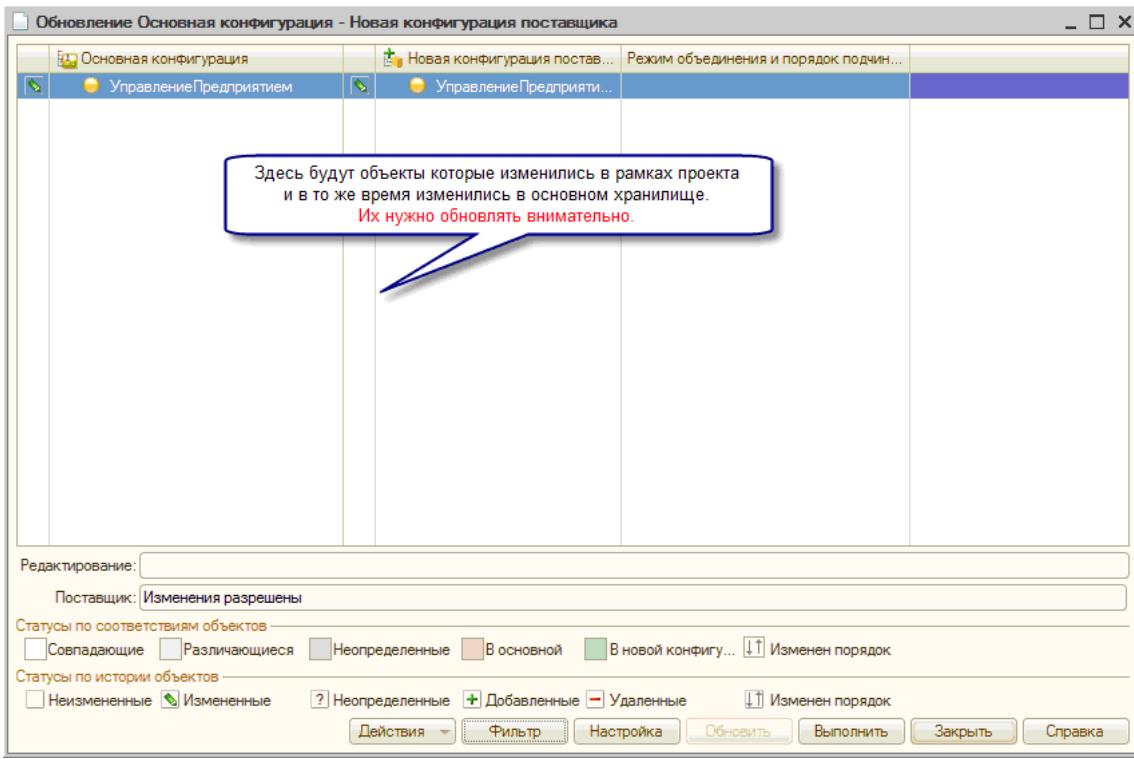
Для того чтобы обновить ХТП до состояния ОХ необходимо выполнить следующее:

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 в текстовом формате и формате табличного документа.

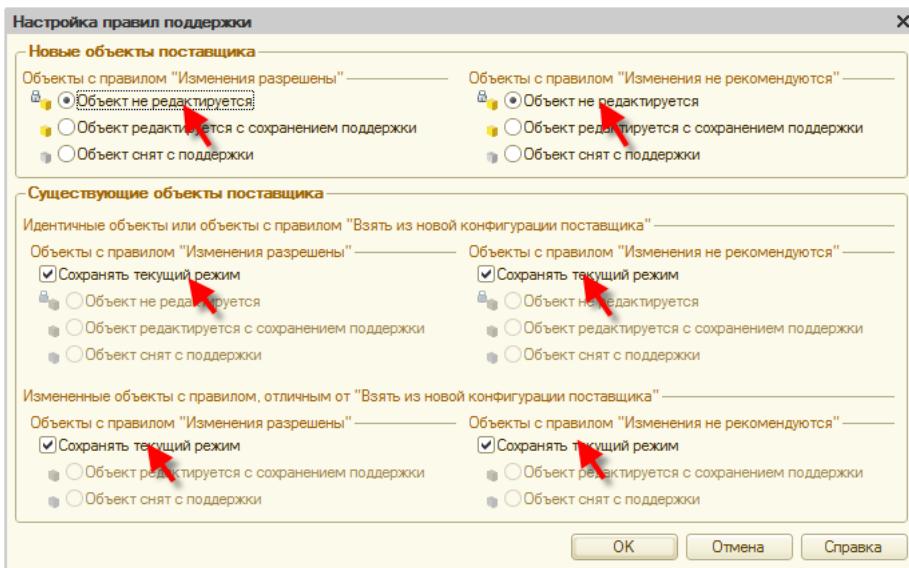


5. Обновить конфигурацию (Конфигурация – Поддержка – Обновить конфигурацию – Выбор файла обновления – указать файл поставки конфигурации созданный на шаге 2).

В появившемся окне сравнения и объединения конфигураций нажать кнопку "Фильтр" и установить флажок Показывать только дважды измененные свойства".

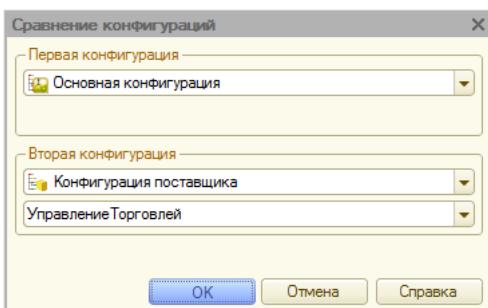


6. В диалоге, который появляется при нажатии на кнопку «Выполнить» окна сравнения и объединения конфигураций, для новых объектов поставщика нужно установить правило «Объект не редактируется» - как для объектов с правилом «Изменения разрешены» так и для объектов с правилом «Изменения не рекомендуются», 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 OC Linux и macOS

Область применения: управляемое приложение, обычное приложение.

#std723

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

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 **СОМОбъект**) 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 ras и rac 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](#).

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 регистра-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 разделять 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 **ЭтоЛinuxКлиент**, **ЭтоМacOSКлиент** и **ЭтоВindowsКлиент** из общих модулей **ОбщегоНазначения** и **ОбщегоНазначенияКлиент**.

См. также

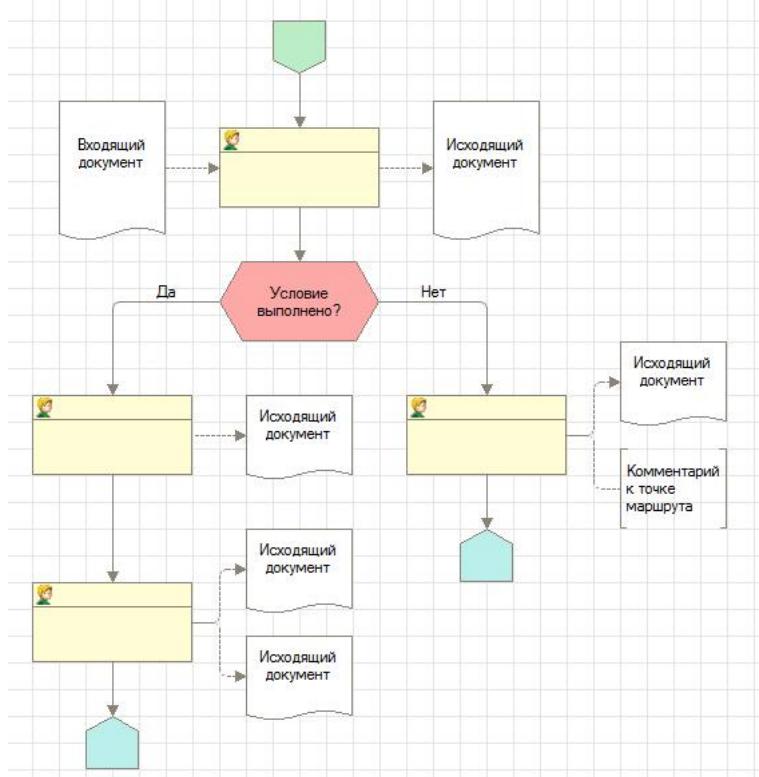
- [Общие требования к конфигурации](#)
- [Глава "35. Особенности разработки кроссплатформенных прикладных решений"](#) документации по платформе 1С:Предприятие

Оформление карты маршрута бизнес-процесса

Область применения: управляемое приложение, обычное приложение.

#std480

Пример оформления бизнес-процесса:

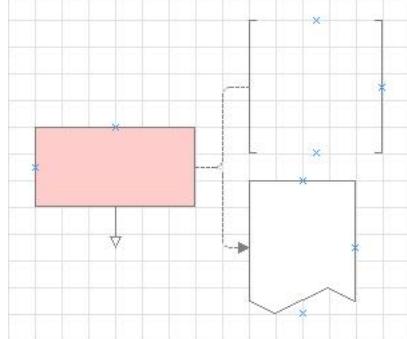


1. Общие рекомендации

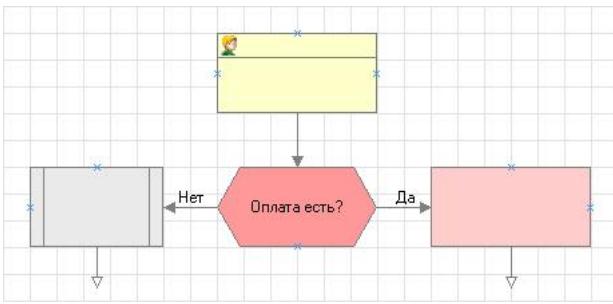
- 1.1 Бизнес-процессы рисуются **сверху - вниз**. Другая ориентация (например, слева направо) допустима, если позволяет изобразить бизнес-процесс наилучшим образом.
- 1.2 Используется стандартная **сетка** – шаг 20 точек, нарисованная линиями.
- 1.3 Не используются жирные и разноцветные **шрифты**.
- 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 **Ширина** 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, 3x6 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 2x2 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:



- 2.5 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?**, **OK?**). 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

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

#std706

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](#)

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 **ПолноеИмяТипаСтруктуры**, 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.

Искключение составляют роли и подсистемы, для которых автоматически не отслеживаются переименования, и для них требуется в явном виде описать переименования. Подробнее см. [документацию к БСП](#).

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](#)

Требования к установке и обновлению прикладных решений

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

#std731

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](#) 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>\1С\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=Управление Торговлей
Version=11.1.3.6
AppVersion=8.3
[Config1]

```

```
Catalog=1C:Управление торговлей/Управление торговлей  
Destination=1C\Trade  
Source=1cv8.cf  
[Config2]  
Catalog=1C: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

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

#std759

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.

Ограничение на использование модальных окон и синхронных вызовов.

См. также

- [Общие требования к конфигурации](#)

Общие сведения о выпуске конфигураций

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

#std484

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](#)

Нумерация редакций и версий

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

#std483

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](#)

Заполнение свойств конфигурации информацией о выпуске

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

#std482

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](#).

7. Авторские права. Указывается строка вида:

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

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

Вместо "xxxx-xxxx" 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>

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

<http://v8.1c.ru/<короткое имя>/>

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](#).

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

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

Общие сведения об организации хранения данных

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

#std683

Методическая рекомендация (полезный совет)

1. При проектировании системы одной из задач является выбор типов объектов метаданных 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:

Хранение данных*



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

На схеме выделяются следующие блоки:

- 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.
- 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.
- 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 хранения информации см. в книге «[Профессиональная разработка в системе 1С Предприятие 8](#)» 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 сведений	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 изменяющиеся реквизиты произвольных данных	Необходимо хранить ссылки на другие объекты и значения параметров, характеризующие события	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.

- Если требуется хранение плана счетов для организации учета по [принципам двойной записи](#), то используется специализированный объект метаданных «[План счетов](#)»
- Если требуется хранение перечня видов расчета для организации [учета начислений и удержанний](#), то используется специализированный объект метаданных «[План видов расчета](#)»
- Если требуется хранить список характеристик (свойств), причем состав самого списка, тип характеристик, их состав определяются пользователем, то используется объект метаданных «[План видов характеристик](#)»

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](#)»
5. Если необходимо определить фиксированный список значений не редактируемый пользователем, без каких-либо дополнительных реквизитов, то используется объект метаданных «[Перечисление](#)»
6. В остальных случаях, как правило, используется объект метаданных «[Справочник](#)»

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](#)»
2. Если требуется регистрировать в системе возникновение и ход регулярного процесса, состоящего из последовательности действий (событий), то используется объект метаданных «[Бизнес-процесс](#)». Для учета событий, действий в рамках процесса используется объект метаданных «[Задача](#)»

3. В остальных случаях, как правило, используется объект метаданных «[Документ](#)»

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](#), то используется специализированный объект метаданных «[Регистр бухгалтерии](#)»
2. Если требуется хранение результатов расчета [учета начислений и удержаний](#), то используется специализированный объект метаданных «[Регистр расчета](#)»

3. Если требуется хранение изменений показателей – приход и расход, получение остатков и оборотов за период, то используется объект метаданных «[Регистр накопления](#)».

4. Во всех остальных случаях используется объект метаданных «[Регистр сведений](#)».

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

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

#std541

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

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

#std474

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 распространяется 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: [Тексты](#)

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 чых 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](#)

2.2. Исключение из этого правила составляют [объекты метаданных с префиксом Удалить](#).

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](#).

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 выполнении 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](#)
- [Подсказка и проверка заполнения](#)
- [Реквизит "Комментарий" у документов](#)

Подсказка и проверка заполнения

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

#std478

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, автоматически генерируемые конфигуратором при создании элементов управления (групп) на формах, нужно не забывать очищать. Например, «Группа шапка» и пр.

См. также: [Тексты](#)

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](#).

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](#)

Использование кодов (номеров) объектов конфигурации

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

#std473

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.

2. 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.

3. 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](#).

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.

4. 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

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

#std432

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 реквизите 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** – 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 потеряна 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.ерf](#)

См. также

- [Формирование печатных форм](#)

Ограничения на использование реквизитов составного типа

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

#std728

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 снижается. Это обусловлено особенностями физического хранения реквизитов составных типов в колонках таблиц СУБД. См. [Особенности хранения составных типов данных](#) (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.

1.3. Исключение могут составлять таблицы, в которых заведомо мало данных (до 1000 записей).

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), действительно рассчитанные на работу с произвольными ссылочными объектами.

Распространенные сложности из-за избыточного использования «сильно» составных типов:

- При обращении «через точку» к реквизиту такого типа без [ВЫРАЗИТЬ](#), выполняется неявное соединение со всеми таблицами, входящими в составной тип. Это приводит к существенной деградации производительности.
- Избыточная реструктуризация при удалении ссылочного объекта метаданных (например, значения перечисления или точки маршрута бизнес-процесса, которые также входят в тип [ЛюбаяСсылка](#)).
- Неоправданно усложняются алгоритмы, которые избыточно должны рассчитывать на широкий набор объектов метаданных тех типов, для которых механизм не требуется в силу прикладной задачи.
- Так же усложняется анализ работы механизма для внедренцев и сторонних разработчиков.
- При вводе значений в поля форм неудобно выбирать из очень большого списка типов конфигурации (когда реально нужно, например, 3-5 типов).
- Избыточный анализ и блокировки таблиц при удалении помеченных объектов, когда выполняется поиск ссылок на удаляемый.

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 [определенные типы](#).

Исключение могут составлять типы ведущих измерений регистров универсальных (библиотечных) механизмов, рассчитанных на работу с произвольными ссылочными объектами. В этом случае при необходимости сузить состав определяемого типа в конфигурации придется пересоздавать библиотечный регистр, что недопустимо. Поэтому здесь использование составных типов [ЛюбаяСсылка](#), [СправочникСсылка](#), [ДокументСсылка](#) и аналогичных оправданно.

См. также:

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

Требования к проведению документов

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

#std603

1. Документы предназначены для ввода первичной информации, связанной с регистрацией событий, действующих 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 сложной 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](#).

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](#) 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](#)
- [Порядок записи движений документов](#)
- [Самодостаточность регистров](#)

Использование активности движений

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

#std633

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](#)
- [Порядок записи движений документов](#)

Самодостаточность регистров

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

#std477

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](#)
- [Разработка планов обмена с отборами](#)

Реквизит «Комментарий» у документов

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

#std531

Методическая рекомендация (полезный совет)

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" \(8.2\)](#)
- [Поля "Ответственный" и "Комментарий" \(8.3\)](#)

Удаление устаревших объектов метаданных из конфигурации

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

#std534

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.

кроме ролей **ПолныеПрава** и **АдминистраторСистемы**), 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

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

#std632

Следует выполнять запись константы вне транзакций, т.к. ее запись может стать «узким» местом при конкурентной 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»](#).

Вместе с тем, недопустимо решать проблему блокировок констант другими методами, в частности, тотальным кешированием констант в [общих модулях с повторным использованием возвращаемых значений](#).

Работа с неактуальными (недействительными) объектами

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

#std638

Методическая рекомендация (полезный совет)

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 выбирать 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:

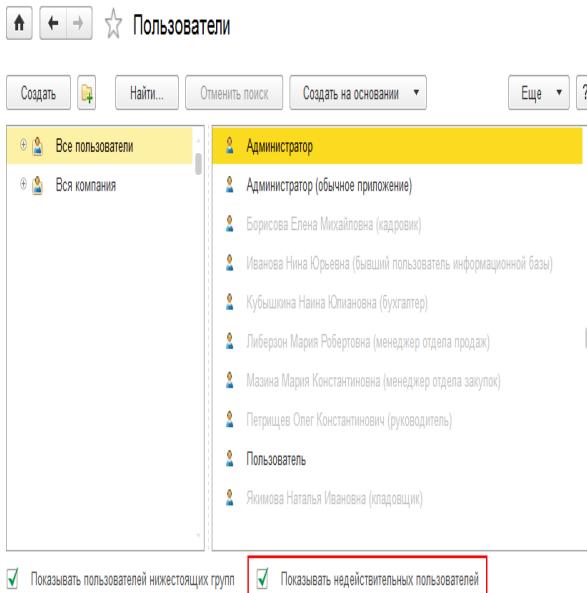
- если требуется выбирать недействующих пользователей, указываем в свойстве реквизита **«Параметры выбора»** значение **Отбор.Недействителен(Ложь)**,
- а если не требуется ограничивать выбор вообще, то оба значения - **Истина** и **Ложь**.

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 сотруднику, вернувшуюся из декретного отпуска).

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



Использование предопределенных элементов

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

#std697

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**, предопределенный 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](#) (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 изменениях 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 ссылается 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](#)) 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:

```
... = ПланыВидовРасчета.Начисления.ОкладПоЧасам;
... = ПредопределенноеЗначение("ПланыВидовРасчета.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 ("ПланыВидовРасчета.Начисления.ОкладПоЧасам");
```

См. также

- [Получение предопределенных значений на клиенте](#)

Обработчик события ПередЗаписью

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

#std464

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, связанные 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](#)»

2. Все действия в процедуре-обработчике события **ПередЗаписью** должны выполняться после [проверки на ОбменДанными.Загрузка](#).

Обработчик события ПриЗаписи

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

#std465

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](#).

См. также

- [Раздел «Проверки, выполняемые в и вне транзакции записи объекта»](#) статьи «[Обработчик события ОбработкаПроверкиЗаполнения](#)»

Обработчик события ПередУдалением

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

#std752

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](#).

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

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

#std466

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

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

#std463

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](#)

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 ИИНСоответствуетТребованиям(ИИН) Тогда
    Сообщение = Новый СообщениеПользователю();
    Сообщение.Текст = НСтр("ги = 'ИИН задан неверно.'");
    Сообщение.Поле = "ИИН";
    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 ИИНСоответствуетТребованиям(ИИН) Тогда
    Сообщение = Новый СообщениеПользователю();
    Сообщение.Текст = НСтр("ги = 'ИИН задан неверно.'");
    Сообщение.Поле = "ИИН";
    Сообщение.УстановитьДанные(ЭтотОбъект);
    Сообщение.Сообщить();
    Отказ = Истина;
    НепроверяемыеРеквизиты.Добавить("ИИН");
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 внутрьтранзакционных 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 проверку 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**, **ПриЗаписи**, **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, несмотря на то, что документ "Инвентаризация товаров" изменяет остатки товаров без проверок, эта ситуация допустима ввиду 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

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

#std396

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 ТипЗнач(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;
    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 = ТипЗнач(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
- Обращение из кода к автоматически формируемым элементам управления формы

Обработчики событий ОбработкаПолученияПредставления и ОбработкаПолученияПолейПредставления

Область применения: управляемое приложение, мобильное приложение.

#std746

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.ПодставитьПараметрыВСтрочку(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.

Разработка планов обмена с отборами.

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](#), 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](#).

См. также

- [Получение представлений для ссылочных значений в табличном документе](#)
- [Пользовательские представления объектов](#)

Использование признака ОбменДанными.Загрузка в обработчиках событий объекта

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

#std773

1. Все действия в процедурах-обработчиках событий [ПередЗаписью](#), [ПриЗаписи](#), [ПередУдалением](#) 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 = 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](#), 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.

Общие требования к регламентным заданиям

Область применения: управляемое приложение, обычное приложение.

#std540

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 флагом **Использование**), которое должно выполняться только в том случае, если установлена ФО **ИспользоватьПочтовыйКлиент**.

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

Правильно: снять флагок **Использование** и управлять использованием регламентного задания в зависимости от включения/выключения функциональной опции.

Если в конфигурации используется подсистема «Регламентные задания» Библиотеки стандартных 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.ПриНачалеШага(Метаданные.РегламентногоЗадания());
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 администрации 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 помечать вызов:

```
ОбщегоНазначения.ПриНачалеШага(Метаданные.РегламентныеЗадания.<ИмяЗадания>);
```

Первый параметр при этом заполнять обязательно.

См. также

- [Настройка расписания регламентных заданий](#)
- [Запуск регламентных заданий](#)
- [Регламентные задания: требования по локализации](#)
- [Ограничения на регламентные задания при работе в режиме сервиса](#)

Настройка расписания регламентных заданий

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](#)
- [Запуск регламентных заданий](#)

Запуск регламентных заданий

Методическая рекомендация (полезный совет)

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:Предприятие**;
- 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:

```
&На Сервере
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("гру = '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](#)

- [Предопределенные регламентные задания](#)
- [Ограничения на регламентные задания при работе в режиме сервиса](#)

Ограничения на регламентные задания при работе в режиме сервиса

Область применения: управляемое приложение.

#std760

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 разделяемые 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 разделяющей 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 задание **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.

Оформление текстов запросов

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

#std437

1. Все ключевые слова языка запросов пишутся заглавными буквами.

Методическая рекомендация (полезный совет)

2. Рекомендуется указывать и необязательные конструкции запроса, прежде всего - явно назначать псевдонимы полям, в целях повышения наглядности текста запроса и "устойчивости" использующего его кода. Например, если в алгоритме используется запрос с полем, объявленным как

Касса.Валюта

при изменении имени реквизита нужно будет также изменить и код, осуществляющий обращение по имени свойства **Валюта** к выборке из результата запроса. Если же поле будет объявлено как

Касса.Валюта КАК Валюта

то изменение имени реквизита приведет только к изменению текста запроса.

2а. Особенно внимательно следует относиться к автоматически присваиваемым псевдонимам для полей – реквизитов других полей, типа "... Касса.Валюта.Наименование...". В приведенном выше примере поле получит автоматический псевдоним **ВалютаНаименование**, а не **Наименование**.

2б. Следует обязательно указывать ключевое слово **КАК** перед псевдонимом поля источника.

3. Текст запроса должен быть структурирован, не следует писать запрос в одну строку, даже короткий. Текст запроса должен быть нагляден, поскольку это существенно улучшает его понимание другими разработчиками.

4. В запросы, сложные для понимания, в которых используются вложенные запросы, объединения или соединения рекомендуется вставлять комментарии. Комментарии, например, могут объяснять для получения каких данных используется та или иная таблица в соединении или объединении.

При этом необходимо иметь в виду, что при использовании конструктора запросов, все комментарии в запросе удаляются автоматически без предупреждения.

5. При создании объекта **Запрос** рекомендуется указывать комментарии, для получения какой информации или каких иных целей будет использован данный запрос.

6.1 При программной "сборке" текста запроса рекомендуется комментировать все этапы его сборки.

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

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

Типичные случаи программной модификации текста запроса

Изменение имени поля выборки или таблицы

Неправильно

ТекстЗапроса =

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

Правильно

ТекстЗапроса =

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

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

или аналогично для имени таблицы

ТекстЗапроса =

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

или еще один вариант для имени таблицы

ТекстЗапроса =

```
"ВЫБРАТЬ
| Номенклатура.Наименование КАК НаименованиеТовара ,
| ЕСТЬNULL(ТаблицаОстатков.ВНаличииОстаток,0) КАК ОстатокТовара
| ИЗ
| Справочник.Номенклатура КАК Номенклатура
| ЛЕВОЕ СОЕДИНЕНИЕ #ТаблицаОстатков КАК ТаблицаОстатков
| ПО Номенклатура.Ссылка= ТаблицаОстатков.Номенклатура";
```

Если ИспользуетсяАдресноеХранение Тогда

ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "#ТаблицаОстатков", "РегистрНакопления.ТоварыВЯчейках.Остатки");

Иначе

ТекстЗапроса = СтрЗаменить(ТекстЗапроса , "#ТаблицаОстатков", "РегистрНакопления.ТоварыНаСкладах.Остатки");

КонецЕсли;

Конкатенация нескольких текстов запросов в пакет

Неправильно

ТекстЗапроса = " ";

Если ИспользоватьУпаковки Тогда

ТекстЗапроса =

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

КонецЕсли;

ТекстЗапроса = ТекстЗапроса +

```
"ВЫБРАТЬ
| Номенклатура.Ссылка КАК Ссылка
```

```

| ИЗ
| Справочник.Номенклатура КАК Номенклатура";
Правильно
ТекстЗапроса = " ";
Если ИспользоватьУпаковки Тогда
ТекстЗапроса =
"ВЫБРАТЬ
| Упаковки.Ссылка КАК Ссылка
|ИЗ
| Справочник.Упаковки КАК Упаковки";
ТекстЗапроса = ТекстЗапроса +
"
|;
|//////////";
КонецЕсли;
ТекстЗапроса = ТекстЗапроса +
"ВЫБРАТЬ
| Номенклатура.Ссылка КАК Ссылка
|ИЗ
| Справочник.Номенклатура КАК Номенклатура";
Или
Разделитель =
";
|//////////";
ТекстыЗапросовПакета = Новый Массив;
ТекстЗапроса =
"ВЫБРАТЬ
| Упаковки.Ссылка КАК Ссылка
|ИЗ
| Справочник.Упаковки КАК Упаковки";
ТекстыЗапросовПакета.Добавить(ТекстЗапроса);
ТекстЗапроса =
"ВЫБРАТЬ
| Номенклатура.Ссылка КАК Ссылка
|ИЗ
| Справочник.Номенклатура КАК Номенклатура ";
ТекстыЗапросовПакета.Добавить(ТекстЗапроса);
ТекстЗапроса = СтрСоединить(ТекстыЗапросовПакета, Разделитель);

```

См. также

- [Запросы, динамические списки и отчеты на СКД: требования по локализации](#)

Многократное выполнение однотипных запросов

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

#std436

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

Правильно:

```

// БанкиДляОбработки - содержит массив банков, счета в которых необходимо обработать

ОбщийЗапрос = Новый Запрос(
    | ВЫБРАТЬ
    | БанковскиеСчета.Ссылка КАК Счет
    |ИЗ
    | Справочник.БанковскиеСчета КАК БанковскиеСчета
    |ГДЕ
    | БанковскиеСчета.Банк В(&БанкиДляОбработки"));

ОбщийЗапрос.УстановитьПараметр("БанкиДляОбработки", БанкиДляОбработки);
ВыборкаСчетов = ОбщийЗапрос.Выполнить().Выбрать();
Пока ВыборкаСчетов.Следующий() Цикл
    ОбработатьСчетаВБанке(ВыборкаСчетов.Счет);
КонецЦикла;

```

Неправильно:

```

// БанкиДляОбработки - содержит массив банков, счета в которых необходимо обработать

ЧастныйЗапрос = Новый Запрос(
    | ВЫБРАТЬ
    | БанковскиеСчета.Ссылка КАК Счет
    |ИЗ
    | Справочник.БанковскиеСчета КАК БанковскиеСчета
    |ГДЕ
    | БанковскиеСчета.Банк = &Банк");

Для каждого Банк Из БанкиДляОбработки Цикл
    ЧастныйЗапрос.УстановитьПараметр("Банк", Банк);
    ВыборкаСчетов = ЧастныйЗапрос.Выполнить().Выбрать();
    Пока ВыборкаСчетов.Следующий() Цикл
        ОбработатьСчетаВБанке(ВыборкаСчетов.Счет);

```

КонецЦикла;
КонецЦикла;

Проверка на пустой результат выполнения запроса

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

#std438

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

Неправильно:

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

Правильно:

```
Возврат НЕ Запрос.Выполнить().Пустой()
```

Методическая рекомендация (полезный совет)

2. В то же время если требуется выбирать (или выгрузить) результат запроса, то предварительный вызов метода **Пустой** не требуется. Например, вместо:

```
РезультатЗапроса = Запрос.Выполнить();
Если НЕ РезультатЗапроса.Пустой() Тогда // избыточный вызов
    Выборка = РезультатЗапроса.Выбрать();
    Пока Выборка.Следующий() Цикл
    ...
    ...
```

правильно:

```
Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
    ...
    ...
```

Ограничение на использование конструкции "ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ" в запросах

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

#std435

Методическая рекомендация (полезный совет)

1.1. При разработке текстов запросов следует иметь в виду, что при работе в клиент-серверном варианте, когда в качестве СУБД используется **PostgreSQL**, производительность выполнения запросов с конструкцией **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** значительно снижается. В особенности это касается случаев, когда в запросе встречаются две и более таких конструкций.

Поэтому в общем случае не рекомендуется использовать конструкцию **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** в запросах. И в тех случаях, где это возможно, рекомендуется переписать текст исходного запроса без использования этой конструкции.

Например, следующий запрос:

```
ВЫБРАТЬ
    ЕСТЬNULL(ПланПродаж.Номенклатура, ФактическиеПродажи.Номенклатура) КАК Номенклатура,
    ЕСТЬNULL(ПланПродаж.Сумма, 0) КАК СуммаПлан,
    ЕСТЬNULL(ФактическиеПродажи.Сумма, 0) КАК СуммаФакт
ИЗ
    ПланПродаж КАК ПланПродаж
    ПОЛНОЕ СОЕДИНЕНИЕ ФактическиеПродажи КАК ФактическиеПродажи
    ПО ПланПродаж.Номенклатура = ФактическиеПродажи.Номенклатура
```

может быть реализован без конструкции **ПОЛНОЕ [ВНЕШНЕЕ] СОЕДИНЕНИЕ** следующим образом:

```
ВЫБРАТЬ
    ПланФактПродаж.Номенклатура КАК Номенклатура,
    СУММА(ПланФактПродаж.СуммаПлан) КАК СуммаПлан,
    СУММА(ПланФактПродаж.СуммаФакт) КАК СуммаФакт
ИЗ
```

```
(ВЫБРАТЬ
    ПланПродаж.Номенклатура КАК Номенклатура,
    ПланПродаж.Сумма КАК СуммаПлан,
    0 КАК СуммаФакт
ИЗ
    ПланПродаж КАК ПланПродаж
```

ОБЪЕДИНИТЬ ВСЕ

```
ВЫБРАТЬ
    ФактическиеПродажи.Номенклатура,
    0,
    ФактическиеПродажи.Сумма
ИЗ
    ФактическиеПродажи КАК ФактическиеПродажи) КАК ПланФактПродаж
```

```
СГРУППИРОВАТЬ ПО
    ПланФактПродаж.Номенклатура
```

1.2. Исключение составляют случаи, когда текст исходного запроса не может быть переписан без использования конструкции **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** по объективным причинам. Следует иметь в виду, что при выполнении данной конструкции на СУБД **PostgreSQL** она автоматически заменяется платформой **1С:Предприятие** на эквивалентную, которая может быть исполнена в СУБД **PostgreSQL**. При этом сохраняются все атрибуты запроса, такие как модификаторы **ПЕРВЫЕ**, **РАЗЛИЧНЫЕ**, а также **УПОРЯДОЧИТЬ ПО**. В таких случаях не следует "механически" заменять конструкцию **ПОЛНОЕ ВНЕШНЕЕ СОЕДИНЕНИЕ** только с той целью, чтобы от нее избавиться в тексте запроса.

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

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

См. также

- [Общие требования к конфигурации](#)

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

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

#std434

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

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

Правильно:

```
ВЫБРАТЬ
ПоступлениеТоваровУслуг .Ссылка
ИЗ
Документ.ПоступлениеТоваровУслуг КАК ПоступлениеТоваровУслуг
```

ОБЪЕДИНИТЬ ВСЕ

```
ВЫБРАТЬ
РеализацияТоваровУслуг .Ссылка
ИЗ
Документ.РеализацияТоваровУслуг КАК РеализацияТоваровУслуг
```

Неправильно:

```
ВЫБРАТЬ
ПоступлениеТоваровУслуг.Ссылка
ИЗ
Документ.ПоступлениеТоваровУслуг КАК ПоступлениеТоваровУслуг
```

ОБЪЕДИНИТЬ

```
ВЫБРАТЬ
РеализацияТоваровУслуг.Ссылка
ИЗ
Документ.РеализацияТоваровУслуг КАК РеализацияТоваровУслуг
```

Упорядочивание результатов запроса

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

#std412

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

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

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

Вероятность возникновения разных результатов при выполнении одинаковых действий повышается

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

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

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

Неправильно:

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

УПОРЯДОЧИТЬ ПО
КоличествоОстаток

Правильно:

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

УПОРЯДОЧИТЬ ПО
КоличествоОстаток

См. также: [Особенности работы с различными СУБД](#)

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

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

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

См. также: [Особенности сортировки в таблице значений](#)

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

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

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

Совместное использование с конструкцией РАЗЛИЧНЫЕ

2. Если в запросе используется конструкция **РАЗЛИЧНЫЕ**, упорядочивание следует выполнять только по полям, включенными в выборку (в секции **ВЫБРАТЬ**).

Данное требование связано со следующей особенностью выполнения запросов: в поля выборки неявно включаются поля упорядочивания, что в свою очередь может привести к появлению в результате запроса нескольких строк с одинаковыми значениями полей выборки.

Ограничения на использование конструкции АВТОУПОРЯДОЧИВАНИЕ

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

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

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

Округление результатов арифметических операций в запросах

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

#std535

Методическая рекомендация (полезный совет)

1. Если в операции деления заранее известны порядки числителя и знаменателя, то следует по возможности избегать выполнения деления числа заведомого маленького порядка на число заведомо большого порядка. Например, вместо:

0.02 / 28346 * 9287492

правильно:

0.02 * 9287492 / 28346

2. При выполнении арифметических операций в запросах к базе данных платформа **1С:Предприятия** поддерживает точность вычислений до 8 разрядов дробной части. Однако, из-за особенностей работы различных СУБД в некоторых ситуациях точность результатов может отличаться от 8. Более подробно о вычислении разрядности результатов можно почитать в статье ИТС [Разрядность результатов выражений и агрегатных функций в языке запросов](#).

Если точность результата выполнения запроса к базе данных, содержащего

- арифметические операции деления,
- агрегатные функции **СРЕДНЕЕ**,
- арифметические операции умножения, если каждый из множителей может иметь дробную часть,

различается на различных СУБД, то рекомендуется к операндам и/ или результатам этих операций применять оператор явного приведения разрядности и точности числовых данных:

ВЫРАЗИТЬ(. . . КАК Число(m, n))

Оператор **ВЫРАЗИТЬ** следует применять к операндам, если на какой-нибудь СУБД точность получаемого результата недостаточна. Например, требуется 10 разрядов после запятой, а получается 6.

При этом указанная общая разрядность операндов должна быть минимальной, но не меньше той, которая достаточна для представления значений каждого из операндов. Неоправданное завышение разрядности может привести к потере точности последующих вычислений и несколько снизить скорость выполнения запроса.

Важно иметь в виду, что на разных СУБД имеются различные ограничения на максимальную разрядность десятичных чисел. Самое жесткое ограничение - это 31 разряд в целой и дробной частях. Чем меньшее значение разрядности будет указано для операндов, тем выше сможет быть точность результата. Например, если в результате требуется не менее 10 разрядов дробной части, первый операнд заведомо помещается в 15 разрядов целой части, а второй операнд заведомо помещается в 5 знаков целой части, то выражение может быть записано так:

**ВЫБРАТЬ
ВЫРАЗИТЬ(Таблица.Множитель * Таблица.Числитель КАК Число(25, 10)) / ВЫРАЗИТЬ(Таблица.Знаменатель КАК Число(15, 10)) КАК Результат
ИЗ Таблица КАК Таблица**

Оператор **ВЫРАЗИТЬ** следует применять к результату, если точность вычислений на всех СУБД достаточна, но на некоторых она больше, а на других меньше. При этом указанная общая разрядность результата должна быть минимальной, но не меньше той, которая достаточна для представления значений результата. Если в приведенном примере известно, что **Знаменатель** не может быть меньше 0.00001, то для представления результата достаточно 20 разрядов целой части. В этом случае выражение может быть записано так:

**ВЫБРАТЬ
ВЫРАЗИТЬ(Таблица.Множитель * Таблица.Числитель / Таблица.Знаменатель КАК Число(30, 10)) КАК Результат
ИЗ Таблица КАК Таблица**

Иногда может быть целесообразно выполнить приведения к требуемой точности как операндов, так и результата. Например:

**ВЫБРАТЬ
ВЫРАЗИТЬ(ВЫРАЗИТЬ(Таблица.Множитель * Таблица.Числитель КАК Число(25, 10)) / ВЫРАЗИТЬ(Таблица.Знаменатель КАК Число(15, 10)) КАК
Число(30, 10)) КАК Результат
ИЗ Таблица КАК Таблица**

Особенности использования в запросах оператора ПОДОБНО

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

#std726

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

Допустимо:

Реквизит ПОДОБНО "123%"

Недопустимо:

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

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

Например, вместо:

```
Запрос = Новый Запрос("ВЫБРАТЬ Товары.Ссылка ИЗ Справочник.Товары КАК Товары ГДЕ Товары.СтранаПроисхождения.Наименование ПОДОБНО &ШаблонНазванияСтраны + "_"");
```

```
Запрос.УстановитьПараметр("ШаблонНазванияСтраны", "ЧА_");
```

Необходимо использовать:

```
Запрос = Новый Запрос("ВЫБРАТЬ Товары.Ссылка ИЗ Справочник.Товары КАК Товары ГДЕ Товары.СтранаПроисхождения.Наименование ПОДОБНО &ШаблонНазванияСтраны");
```

```
Запрос.УстановитьПараметр("ШаблонНазванияСтраны", "ЧА_");
```

Данное требование продиктовано необходимостью переносимости прикладных решений на различные СУБД.

См. также

- [Общие требования к конфигурации](#)
- [Оформление текстов запросов](#)

Псевдонимы источников данных в запросах

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

#std758

именам переменных в коде.

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

Неправильно:

```
ВЫБРАТЬ Таблица1.Ссылка КАК Товар,  
ЕстьNULL(Таблица2.КоличествоОстаток, 0) КАК Остаток  
ИЗ Справочник.Номенклатура КАК Таблица1  
ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки КАК Таблица2  
ПО Таблица1.Ссылка = Таблица2.Номенклатура
```

Правильно:

```
ВЫБРАТЬ ВсяНоменклатура.Ссылка КАК Товар,  
ЕстьNULL(ОстаткиНаСкладах.КоличествоОстаток, 0) КАК Остаток  
ИЗ Справочник.Номенклатура КАК ВсяНоменклатура  
ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах.Остатки КАК ОстаткиНаСкладах  
ПО ВсяНоменклатура.Ссылка = ОстаткиНаСкладах.Номенклатура
```

В частности не рекомендуется использовать имена классов объектов метаданных ("Справочник", "Документ" и т.д.), т.к. обычно такой псевдоним не будет описывать назначение источника в конкретном запросе.

2. В ряде случаев при разработке универсальных механизмов, рассчитанных на работу с произвольными таблицами данных, или при написании универсальных запросов, когда вместо источника данных при исполнении кода подставляется имя конкретной таблицы, допустимо использование универсальных псевдонимов.

Пример:

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

Общие требования по разработке оптимальных запросов

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

#std729

Методическая рекомендация (полезный совет)

Прежде чем перейти к более продвинутым методам оптимизации запросов, необходимо убедиться, что сам запрос – адекватен решаемой задаче.

1.1. Следует минимизировать объем выборки таким образом, чтобы выбирать ровно те данные, которые требуются для решения задачи.

Например, если нужно получить значения конкретных полей, не следует выбирать все поля «на всякий случай» с помощью конструкции ВЫБРАТЬ * ИЗ ...

Вместо выборки большого объема данных для их последующей обработки (свертка, сортировка, проведение вычислений и пр.) на сервере 1С:Предприятие, следует, в первую очередь, ответить на вопрос: «А есть ли возможность переложить эту работу на базу данных, чтобы получить уже готовый результат?»

1.2. Также в большинстве случаев, следует минимизировать и общее количество запросов к СУБД.

См. также: [Многократное выполнение однотипных запросов](#).

2. С другой стороны, не следует пытаться любой ценой перенести выполнение задачи в СУБД. СУБД обычно оптимизирует и выполняет простые запросы более эффективно, чем сложные.

2.1. Следует рассмотреть альтернативные меры:

- по подготовке различных (более простых, частных) текстов запроса в зависимости от предусловий и значений параметров запроса – вместо отправки в СУБД одного большого универсального запроса;
- по более эффективной постобработке данных, выбранных запросом из СУБД, на стороне сервера 1С:Предприятия средствами встроенного языка.

2.2. При разработке запросов нужно быть уверенным, что они использует эффективные планы выполнения запросов. Для сложных запросов СУБД с высокой вероятностью выберет неправильный план выполнения запроса, что особенно актуально для СУБД DB2, PostgreSQL и Oracle.

Поэтому не следует неоправданно усложнять запрос, в первую очередь:

- Не следует добавлять [вложенные запросы](#) только для повышения читаемости.
- Избегать сложных условий соединения и в предложении ГДЕ, в особенности [содержащие подзапросы](#) и конструкции ВЫБОР.
- Использовать в запросе минимально необходимое число таблиц. В зависимости от структуры таблиц, много может быть уже и 5-7 таблиц в одном запросе (время, затрачиваемое оптимизатором СУБД на анализ запроса, растет нелинейно, в итоге получается плохой план выполнения).

Для того чтобы узнать, какой план выполнения запроса выбран оптимизатором СУБД, можно воспользоваться консолью запросов, технологическим журналом или средствами СУБД. Как правило, запрос – сложный и будет плохо выполняться, если в скомпилированном плане выполнения запроса есть timeout warning, который означает, что оптимизатору СУБД не хватило времени на поиск наилучшего плана запроса.

См. также: [Запросы в динамических списках](#)

Несоответствие индексов и условий запроса

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

#std652

Методическая рекомендация (полезный совет)

1.1. Необходимо убедиться в том, что для всех условий, использованных в запросе, имеются подходящие индексы.

Условия используются в следующих секциях запроса:

- ВЫБРАТЬ ... ИЗ ... ГДЕ <условие>
- СОЕДИНЕНИЕ ... ПО <условие>
- ВЫБРАТЬ ... ИЗ <ВиртуальнаяТаблица>(<условие>)
- ИМЕЮЩИЕ <условие>

Для каждого условия должен существовать подходящий индекс. Подходящим является индекс, удовлетворяющий следующим требованиям:

1. Индекс содержит все поля перечисленные в условии;
2. Эти поля находятся в самом начале индекса;
3. Эти поля идут подряд, то есть между ними не «вклиниваются» поля, не участвующие в условии запроса

1.2. Если в структуре базы данных отсутствует индекс, удовлетворяющий всем перечисленным условиям, то для получения результата СУБД будет вынуждена сканировать таблицу или один из ее индексов. Это приведет к увеличению времени выполнения запроса, а также к возможному снижению параллельности системы, поскольку возрастет количество установленных блокировок.

Требования к индексу связаны с физической структурой индекса в СУБД. Эта структура представляет собой дерево значений проиндексированных полей. На первом уровне дерева находятся значения первого поля индекса, на втором – второго и так далее. Такая структура позволяет достичь высокой эффективности при поиске по индексу. Кроме того, она гарантирует отсутствие деградации производительности индекса с ростом количества данных.

Однако, индекс такой структуры, очевидно, может быть использован только строго определенным образом. Сначала необходимо провести поиск по значению первого поля индекса, затем – второго и так далее. Если, например, условие по первому полю индекса не указано, то индекс уже не сможет обеспечить быстрый поиск. Если указано условие по нескольким первым полям индекса, а затем одно или несколько полей индекса не задано, то индекс может быть использован только частично.

2. При создании объекта метаданных 1С:Предприятие автоматически создает индексы, которые должны подходить для работы большинства запросов.

Основные индексы, создаваемые 1С:Предприятием:

- индекс по уникальному идентификатору (ссылке) для всех объектных сущностей (справочники, документы и т.д.);
- индекс по регистратору (ссылке на документ) для таблиц движений регистров, подчиненных регистратору;
- индекс по периоду и значениям всех измерений для итоговых таблиц регистров накопления;
- индекс по периоду, счету и значениям всех измерений для итоговых таблиц регистров бухгалтерии.

3. В тех случаях, когда автоматически созданных индексов недостаточно, можно дополнительно проиндексировать реквизиты объекта метаданных.

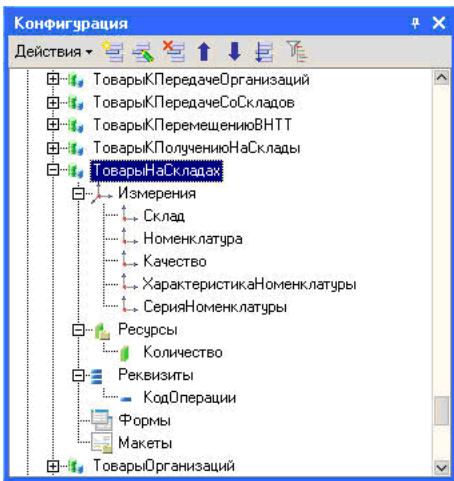
При этом реквизиты справочников и документов рекомендуется индексировать с дополнительным упорядочиванием. Такой индекс будет учитывать упорядочивание по основному представлению объекта, тем самым он будет эффективно использоваться, например, когда в списке установлен отбор по данному реквизиту, а сам список упорядочен по полям основного представления объекта.

Следует иметь в виду, что создание индекса ускоряет процесс поиска информации, но может несколько замедлить процесс ее изменения (добавления, редактирования и удаления). Поэтому индексы следует создавать осознанно и только в том случае, если точно известен запрос, для которого такой индекс необходим. Не следует создавать индексы "на всякий случай" или заведомо избыточные индексы. В частности:

- не следует дополнительно индексировать первое измерение регистра, поскольку для поиска по значению первого измерения подходит основной индекс таблицы итогов, который автоматически создаст платформа;
- не следует создавать индексы по низкоселективным полям. Например, индексировать реквизит типа Булево имеет смысл, только если незначительная часть записей всегда будет иметь одно значение, и в запросах всегда выбираются записи по этому значению.

Примеры

В конфигурации описан регистр накопления ТоварыНаСкладах:



Платформа 1С:Предприятие автоматически создаст для таблицы остатков данного регистра индекс по периоду и всем измерениям в том порядке, в котором они перечислены в конфигураторе.

Рассмотрим несколько примеров запросов и проанализируем, смогут ли они оптимально выполняться при такой структуре данных.

Запрос 1

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Номенклатура = &Номенклатура) КАК ТоварыНаСкладахОстатки";
```

В данном случае нарушено требование 2 раздела 1.1. В условии отсутствует отбор по первому полю индекса (**Склад**). Такой запрос не сможет выполниться оптимально. Для его выполнения серверу СУБД придется перебирать все записи таблицы. Время выполнения этой операции напрямую зависит от количества записей в таблице остатков регистра и может быть очень большим (и будет увеличиваться с ростом количества данных).

Варианты оптимизации:

- Проиндексировать измерение **Номенклатура**
- Поставить измерение **Номенклатура** первым в списке измерений. Следует осторожно использовать этот метод. В конфигурации могут присутствовать другие запросы, которые могут замедлиться в результате этой перестановки.

Запрос 2

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(
|   Качество = &Качество
|   И Склад = &Склад) КАК ТоварыНаСкладахОстатки";
```

В данном случае нарушено требование 3 раздела 1.1. Между измерениями **Склад** и **Качество** в структуре регистра находится измерение **Номенклатура**, которое не задано в условии запроса. Этот запрос так же не сможет выполняться оптимально. При его выполнении СУБД выполнит поиск по первому полю индекса, но затем вынужденно просканирует некоторую его часть. Сканирование может привести к существенному увеличению времени выполнения запроса.

Варианты оптимизации:

- Добавить в запрос условие по измерению **Номенклатура**
- Убрать из запроса условие по измерению **Качество**
- Поменять местами измерения **Номенклатура** и **Качество**. Данную рекомендацию следует использовать, если времена выполнения запроса критическим образом сказываются на работы системы (например, запрос выполняется достаточно часто, или он выполняется не часто, но время его выполнения неприемлимо большое), т.к. это может замедлить выполнение других запросов, присутствующих в конфигурации.

Запрос 3

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество,
| ТоварыНаСкладахОстатки.КоличествоОстаток
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(
|   Номенклатура = &Номенклатура
|   И Склад = &Склад) КАК ТоварыНаСкладахОстатки";
```

В этом случае требования соответствия индекса и запроса не нарушены. Данный запрос будет выполнен СУБД оптимальным способом. Следует обратить внимание на то, что порядок следования условий в запросе не обязан совпадать с порядком следования полей в индексе. Это не является проблемой и будет нормально обработано СУБД.

См. также

- [Индексы таблиц базы данных](#)
- [Эффективные условия запросов](#)

Разыменование ссылочных полей составного типа в языке запросов

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

#std654

1.1. В языке запросов возможно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении **ИЗ**, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся "через точку". Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей "через точку".

Например, в запросе

```
ВЫБРАТЬ
ТоварныеЗапасы.Товар КАК Товар,
ТоварныеЗапасы.Количество КАК Количество,
ТоварныеЗапасы.Товар.Артикул КАК Артикул
ИЗ
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы
...
```

кроме явно указанной в предложении ИЗ таблицы **РегистрНакопления.ТоварныеЗапасы** неявно участвует таблица **Справочник.Товары** для получения значения поля **Артикул**. А в случае использования ограничений доступа на уровне записей (RLS), к запросу добавляются ещё и таблицы, участвующие в RLS к таблице **Справочник.Товары**.

составной тип и может содержать ссылки на несколько таблиц. В таком случае, получение полей других таблиц "через точку" от такого поля составного типа приведет к соединению со всеми таблицами, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Например, в приведенном ниже запросе получение даты регистратора приведет к неявному соединению с таблицами всех документов - регистраторов регистра **ТоварныеЗапасы**

```
ВЫБРАТЬ
...
ТоварныеЗапасы.Регистратор.Дата,
...
ИЗ
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы
...
```

Подобное получение данных "через точку" от ссылочных полей составного типа крайне нежелательно. Каждое исключение из этого правила должно тщательно анализироваться.

2.1. Следует избегать избыточности при создании полей составных ссылочных типов. Необходимо указывать ровно столько возможных типов для данного поля, сколько необходимо. Не следует без необходимости использовать типы "любая ссылка" или "ссылка на любой документ" и т.п.

Вместо этого следует более тщательно проанализировать прикладную логику и назначить для поля ровно те возможные типы ссылок, которые необходимы для решения задачи.

2.2. Для того чтобы избежать запросов с использованием большого числа исходных таблиц следует жертвовать компактностью хранения данных ради производительности и помещать соответствующие данные в исходную таблицу запроса.

Например, в регистре **ТоварныеЗапасы** можно завести реквизит **ДатаРегистратора**, заполнять его при проведении документов и использовать затем в запросах:

```
ВЫБРАТЬ
...
ТоварныеЗапасы.ДатаРегистратора,
...
ИЗ
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы
...
```

Это приведет к дублированию информации и некоторому (незначительному) увеличению ее объема, но может существенно повысить производительность и стабильность работы запроса.

2.3. При необходимости следует жертвовать компактностью и универсальностью кода ради производительности:

- Как правило, для выполнения конкретного запроса в данных условиях не нужны все возможные типы данной ссылки. В этом случае, следует ограничить количество возможных типов при помощи функции **ВЫРАЗИТЬ**.
- Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию **ВЫРАЗИТЬ** тот тип, который необходим при данных условиях.

Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.

Например, неправильно:

```
Запрос.Текст = "ВЫБРАТЬ
| Продажи.Регистратор.Номер,
| Продажи.Регистратор.Дата,
| Продажи.Контрагент,
| Продажи.Количество,
| Продажи.Стоимость
| ИЗ
| РегистрНакопления.Продажи КАК Продажи
| ГДЕ ..."
```

В данном запросе используется обращение к реквизитам регистратора. Регистратор является полем составного типа, которое может принимать значения ссылки на один из 56 видов документов.

SQL-текст этого запроса будет включать 56 левых соединений с таблицами документов. Это может привести к серьезным проблемам производительности при выполнении запроса.

Правильно:

Для решения данной конкретной задачи нет необходимости соединяться со всеми 56 видами документов. Условия запроса таковы, что при его выполнении будут выбраны только движения документов **РеализацияТоваровУслуг** и **ЗаказыПокупателя**. В этом случае можно значительно ускорить работу запроса, ограничив количество соединений при помощи функции **ВЫРАЗИТЬ()**.

```
Запрос.Текст = "ВЫБРАТЬ
| ВЫБОР
| КОГДА Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.РеализацияТоваровУслуг).Номер
| КОГДА Продажи.Регистратор ССЫЛКА Документ.ЗаказПокупателя
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.ЗаказПокупателя).Номер
| КОНЕЦ ВЫБОРА КАК Номер,
| ВЫБОР
| КОГДА Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.РеализацияТоваровУслуг).Дата
| КОГДА Продажи.Регистратор ССЫЛКА Документ.ЗаказПокупателя
| ТОГДА ВЫРАЗИТЬ(Продажи.Регистратор КАК Документ.ЗаказПокупателя).Дата
| КОНЕЦ ВЫБОРА КАК Дата,
| Продажи.Контрагент,
| Продажи.Количество,
| Продажи.Стоимость
| ИЗ
| РегистрНакопления.Продажи КАК Продажи
| ГДЕ
| Продажи.Регистратор ССЫЛКА Документ.РеализацияТоваровУслуг
| ИЛИ Продажи.Регистратор ССЫЛКА Документ.ЗаказыПокупателя";
```

Этот запрос является более громоздким и, возможно, менее универсальным (он не будет правильно работать для других ситуаций - когда возможны другие значения типов регистратора). Однако, при его выполнении будет сформирован SQL запрос, который будет содержать всего два соединения с таблицами документов. Такой запрос будет работать значительно быстрее и стабильнее, чем запрос в его первоначальном виде.

См. также

- [Самодостаточность регистров](#)
- [Ограничения на использование реквизитов составного типа](#)

Ограничения на соединения с вложенными запросами и виртуальными таблицами

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

#std655

1.1. При написании запросов не следует использовать соединения с вложенными запросами. Следует соединять друг с другом только объекты метаданных или временные таблицы. Если запрос использует соединения с вложенными запросами, то его следует переписать с [использованием временных таблиц](#) (не важно с какой стороны соединения находится вложенный запрос), кроме случая, когда вложенный запрос сканирует мало записей.

Если запрос содержит соединения с вложенными запросами, то это может привести к следующим негативным последствиям:

- Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков);
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других - очень медленно;
- Значительная разница по времени выполнения запроса на разных СУБД;
- Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время опять замедлиться.

Пример потенциально опасного запроса, использующего соединение с вложенным запросом:

```
ВЫБРАТЬ ...
ИЗ Документ.РеализацияТоваровУслуг
ЛЕВОЕ СОЕДИНЕНИЕ (
    ВЫБРАТЬ ИЗ РегистрСведений.Лимиты
        ГДЕ ...
        СГРУППИРОВАТЬ ПО ...
    ) ПО ...
```

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае, проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обоих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

1.2. Для вышеупомянутого примера получится следующий пакетный запрос:

```
// Создать менеджер временных таблиц
МенеджерВТ = Новый МенеджерВременныхТаблиц;
Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
// Текст пакетного запроса
Запрос.Текст =
    // Заполняем временную таблицу. Запрос к регистру лимитов.
    | ВЫБРАТЬ ...
    | ПОМЕСТИТЬ Лимиты
    | ИЗ РегистрСведений.Лимиты
    | ГДЕ ...
    | СГРУППИРОВАТЬ ПО ...
    | ИНДЕКСИРОВАТЬ ПО ...;

// Выполняем основной запрос с использованием временной таблицы
ВЫБРАТЬ ...
ИЗ Документ.РеализацияТоваровУслуг
ЛЕВОЕ СОЕДИНЕНИЕ Лимиты
ПО ...;"
```

Переписывание запроса по приведенной выше методике имеет своей целью упростить работу оптимизатору СУБД. В переписанном запросе все выборки, участвующие в соединениях будут представлять собой физические таблицы, и СУБД сможет легко определить размер каждой выборки. Это позволит СУБД гарантированно выбрать самый быстрый из всех возможных планов. Причем, СУБД будет делать правильный выбор независимо ни от каких условий. Переписанный подобным образом запрос будет работать одинаково хорошо на любых СУБД, что особенно важно при разработке тиражных решений. Кроме того, переписанный подобным образом запрос лучше читается, проще для понимания и отладки.

2. Если в запросе используется соединение с виртуальной таблицей языка запросов **1С:Предприятия** (например, **РегистрНакопления.Товары.Остатки**) и запрос работает с неудовлетворительной производительностью, то рекомендуется вынести обращение к виртуальной таблице в отдельный запрос с сохранением результатов во временной таблице (см. пункт 1.1).

3. Следует избегать неявных подзапросов, которые получаются при использовании вложенных соединений:

```
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах
        ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
            ПО ...
        ПО ...
```

Проблема в том, что, по сути, этот запрос аналогичен следующему:

```
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура
    ЛЕВОЕ СОЕДИНЕНИЕ (
        ВЫБРАТЬ ...
        ИЗ РегистрНакопления.ТоварыНаСкладах
            ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
                ПО ...
        ПО ...)
```

Вместо вложенных соединений, как показано выше, следует использовать последовательные соединения:

```
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах
        ПО ...
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
        ПО ...
```

При этом следует понимать, что вложенные и последовательные соединения – это разные запросы, которые могут дать разный результат.

Если вложенное соединение использовано из предположения, что оно аналогично последовательному соединению, то следует просто переписать его на последовательное соединение.

Если вложенное соединение делается осмысленно, то от него следует отказаться, т.к. оно может существенно снизить производительность, как и соединение с подзапросом. Как и в случае с подзапросом, такое соединение можно заменить на соединение с временной таблицей, но лучше вначале подумать, как заменить его на последовательное соединение, т.к. оно будет работать эффективнее временной таблицы.

См. также

- [Использование вложенных запросов в условии соединения](#)
- [Использование временных таблиц](#)

Ограничения на использование вложенных запросов в условии соединения

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

#std656

Не следует использовать вложенные запросы в условии соединения. Это может привести к значительному замедлению запроса и (в отдельных случаях) к его полной неработоспособности на некоторых СУБД. Пример запроса с использованием вложенного запроса в условии соединения:

```
Запрос.Текст = "ВЫБРАТЬ
| ОстаткиТоваров.Номенклатура КАК Номенклатура,
| Цены.Цена КАК ЦенаПрошлогоМесяца
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров
| ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
| ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
| Цены.Период В (
| ВЫБРАТЬ МАКСИМУМ(ЦеныПрошлогоМесяца.Период)
| ИЗ РегистрСведений.Цена КАК ЦеныПрошлогоМесяца
| ГДЕ ЦеныПрошлогоМесяца.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)
| И ЦеныПрошлогоМесяца.Номенклатура = ОстаткиТоваров.Номенклатура
|
| ГДЕ ОстаткиТоваров.Склад = &Склад";
```

В данном случае вложенный запрос в условии соединения используется для получения как бы "реза последних" на конец предыдущего периода. Причем, для каждой номенклатуры период может быть разным.

Подобный запрос рекомендуется переписать с использованием временных таблиц. Например, это можно сделать следующим образом:

```
Запрос.Текст =
// Максимальные даты установки цен в прошлом периоде для данных номенклатур
| ВЫБРАТЬ
| ОстаткиТоваров.Номенклатура КАК Номенклатура,
| МАКСИМУМ(Цены.Период) КАК Период
| ПОМЕСТИТЬ датыПоНоменклатуре
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров
| ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
| ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
| Цены.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)
| СГРУППИРОВАТЬ ПО остаткиТоваров.Номенклатура
| ГДЕ ОстаткиТоваров.Склад = &Склад;

// Выбрать данные по цене за найденный период
| ВЫБРАТЬ
| датыПоНоменклатуре.Номенклатура КАК Номенклатура,
| Цены.Цена КАК ЦенаПрошлогоМесяца
| ИЗ датыПоНоменклатуре
| ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
| ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
| Цены.Период = датыПоНоменклатуре.Период";
```

См. также

- [Ограничения на соединения с вложенными запросами и виртуальными таблицами](#)
- [Использование временных таблиц](#)

Обращения к виртуальным таблицам

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

#std657

1. При использовании виртуальных таблиц в запросах, следует передавать в параметры таблиц все условия, относящиеся к данной виртуальной таблице. Не рекомендуется обращаться к виртуальным таблицам при помощи условий в секции **ГДЕ** и т.п.

Такой запрос будет возвращать правильный (с точки зрения функциональности) результат, но СУБД будет намного сложнее выбирать оптимальный план для его выполнения. В некоторых случаях это может привести к ошибкам оптимизатора СУБД и значительному замедлению работы запроса.

Например, следующий запрос использует секцию **ГДЕ** запроса для выборки из виртуальной таблицы:

```
Запрос.Текст = "ВЫБРАТЬ
| Номенклатура
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки()
| ГДЕ
| Склад = &Склад";
```

Возможно, что в результате выполнения этого запроса сначала будут выбраны все записи виртуальной таблицы, а затем из них будет отобрана часть, соответствующая заданному условию.

Рекомендуется ограничивать количество выбираемых записей на самом раннем этапе обработки запроса. Для этого следует передать условия в параметры виртуальной таблицы.

```
Запрос.Текст = "ВЫБРАТЬ
| Номенклатура
| ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Склад = &Склад)";
```

2.1. При обращении к виртуальной таблице следует передавать в условия наиболее простые конструкции, например, "Измерение = Значение". Не рекомендуется использовать подзапросы и соединения(*) в параметрах виртуальной таблицы, так как это приводит к медленной работе запроса.

* Примечание: как явные соединения в подзапросах, так и неявные – при обращении к полям «через точку» от ссылки и соединения, добавляемые из ограничений доступа к данным (RLS), предусмотренных в ролях конфигурации.

2.2. При необходимости использовать подзапросы рекомендуется соблюдать следующие условия:

- в подзапросе только одна таблица, нет соединений с другими таблицами;
- если в подзапросе таблица табличной части (например, **Документ.Накладная.СписокТоваров**), то не должно быть обращения к реквизитам таблицы-шапки (**Накладная.Проведен**);
- если в подзапросе таблица, у которой могут быть табличные части (например, **Документ.Накладная**), то не должно быть обращений к табличным частям (например, **ГДЕ Документ.Накладная.СписокТоваров.Номенклатура = "1"**);
- если в подзапросе временная таблица, то не должно быть условий (раздела **ГДЕ**);
- если в подзапросе постоянная таблица, то условие (раздел **ГДЕ**) допустимо, только если условие выполняется для 80% (или более) случаев; отсутствие условия означает выполнение для 100% случаев.
- если в подзапросе постоянная таблица, то в ограничениях доступа к данным (RLS) не должно содержаться подзапросов и соединений (допускаются только простые условия вида **ГДЕ Реквизит = Значение, "ГДЕ Истина"**). Например, при использовании стандартных шаблонов RLS, входящих в состав подсистемы «Управление доступом» **Библиотеки стандартных подсистем** к запросу неявно добавляется конструкция **Exists** с несколькими подзапросами и соединениями. В таких случаях следует переписать исходный запрос с использованием временной таблицы или привилегированного режима.

Например, неправильно:

```
... ИЗ
РегистрНакопления.ТоварыОтгрузке.Остатки(
&датаОтгрузки,
&отображениеРаспоряжений
И ДокументОтгрузки.Склад = &Склад           -- неявное соединение «через точку»
ИЛИ ДокументОтгрузки В
(ВЫБРАТЬ
    Распоряжения.Распоряжение КАК ДокументОтгрузки
    ИЗ
    Документ.ЗаданиеНаПеревозку.Распоряжения КАК Распоряжения -- доступ к этому документу ограничен по сложному RLS, который неявно
    добавляет еще пару соединений
    ВНУТРЕННЕЕ СОЕДИНЕНИЕ Документ.ЗаданиеНаПеревозку.СкладыПогрузки КАК СкладыПогрузки
    ПО
    Распоряжения.Ссылка = СкладыПогрузки.Ссылка
    И СкладыПогрузки.Склад = &Склад
    И Распоряжения.Ссылка.Проведен           -- здесь и ниже обращения к реквизитам шапки
    И Распоряжения.Ссылка.Статус В (...))
```

правильно:

```
... ИЗ
РегистрНакопления.ТоварыОтгрузке.Остатки(
&датаОтгрузки,
Склад = &Склад                         -- теперь это реквизит регистра
ИЛИ ДокументОтгрузки В
(ВЫБРАТЬ
    ЗаданияНаПеревозку.Распоряжение
    ИЗ
    ВременнаяТаблицаЗаданийНаПеревозку КАК ЗаданияНаПеревозку)) -- выборка из временной таблицы без условий
```

2.3. В случае, если нужно использовать несколько условий с подзапросами, следует выбрать одно, удовлетворяющее условиям выше и отфильтровывающее максимальное количество записей. Остальные условия следует накладывать на внешний запрос.

Кроме того, в ряде случаев можно обойтись и без перемещения условий на внешний запрос, если применять временные таблицы. Например, вместо условия (неправильно):

Номенклатура В (...) И Характеристика В (...) И Серия В(...)

правильно:

(Номенклатура, Характеристика, Серия) В (ВЫБРАТЬ Номенклатура, Характеристика, Серия ИЗ ВременнаяТаблицаТоваров)

См. также

- [Разыменование ссылочных полей составного типа в языке запросов](#)
- [Использование параметра Условие при обращении к виртуальной таблице](#) (статья на ИТС)
- [Эффективное обращение к виртуальной таблице «Остатки»](#)

Эффективные условия запросов

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

#std658

1. Условия запросов должны быть написаны оптимально с точки зрения производительности, чтобы исключить существенное увеличение длительности выполнения запросов при увеличении объема данных в таблицах.

Поля основного условия в секциях ГДЕ, ПО и виртуальных таблицах должны быть проиндексированы. Основное условие может быть уточнено дополнительным условием, но объединять их следует только по И.

Важно понимать структуру индексов, которые получаются при индексировании полей и учитывать их при построении основного условия (см. [Индексы таблиц базы данных](#), [Несоответствие индексов и условий запроса](#)). Например, при индексировании разных полей одного объекта метаданных создаются разные индексы, а не один в который помещаются все проиндексированные поля и используется основным условием будет только один из них.

Основное условие – это то, что позволяет ограничить объем выборки больше других условий и его составляющие объединены по И.

Дополнительное условие – это то, что объединено с основным условием по И и его составляющие могут быть любой сложности (НЕ, <>, +, -, /, *, функции и т.п.).

Основное условие должно содержать только такие операции, которые позволяют выполнять поиск по индексу:

- для первого и всех используемых полей индекса, кроме последнего, только = и И;
- для последнего или единственного используемого поля индекса допустимо использовать =, >, <, >=, <=, ПОДОБНО, МЕЖДУ, В, ИЛИ (приводимое к В);
- нельзя использовать арифметические операции, функции, отрицания и неравенства.

Для условий в ГДЕ или в виртуальной таблице следует индексировать поля в основной таблице, из которой выполняется выборка.

Для условий в ПО ЛЕВОГО соединения следует индексировать поля в правой таблице.

Для условий в ПО ВНУТРЕННЕГО соединения следует индексировать поля в таблице с большим количеством записей.

Основное условие желательно строить таким образом, чтобы оно использовало индексы, которые автоматически создает платформа.

1.1. Описанные выше требования допустимо не соблюдать, если в таблицах, из которых выполняется выборка, или с которыми выполняется соединение, всегда будет мало данных (менее 1000 записей) или запросы с такими условиями выполняются очень редко.

1.2. Если записей в таблице много и выполнить указанные выше требования невозможно, то можно попробовать:

- преобразовать условия (см. п. 3, п. 4);
- добавить в таблицу заранее вычисляемые индексированные поля, которые заполняются при записи в нее и используются вместо сложного условия;

- если указанные выше рекомендации не помогли, то следует пересмотреть архитектуру решения так, чтобы можно было выполнить эти условия.

2. Оператор ИЛИ

2.1. В основном условии оператор ИЛИ можно использовать только для последнего из используемых или единственного поля индекса, когда оператор ИЛИ можно заменить на оператор В.

ПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле = &Значение1
ИЛИ Таблица.Поле = &Значение2
```

т.к. можно переписать при помощи оператора В (специально переписывать не нужно, можно оставить, как есть):

ГДЕ

```
Таблица.Поле В (&Значения)
```

НЕПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле1 = &Значение1
ИЛИ Таблица.Поле2 = &Значение2
```

нельзя переписать при помощи "В", но можно переписать при помощи "ОБЪЕДИНИТЬ ВСЕ" (каждое поле Поле1 и Поле2 должны быть проиндексированы):

ГДЕ

```
Таблица.Поле1 = &Значение1
```

ОБЪЕДИНИТЬ ВСЕ

ГДЕ

```
Таблица.Поле2 = &Значение1
```

Примечание: заменить ИЛИ на ОБЪЕДИНИТЬ ВСЕ можно не всегда, убедитесь, что результат будет действительно тем же, что и при ИЛИ, перед тем, как применять.

2.2. В дополнительном условии оператор ИЛИ можно использовать без ограничений.

ПРАВИЛЬНО 1:

ГДЕ

```
Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)
И // Дополнительное условие (можно использовать ИЛИ)
(Таблица.Поле2 = &Значение2 ИЛИ Таблица.Поле3 = &Значение3)
```

ПРАВИЛЬНО 2:

ГДЕ

```
(Таблица.Поле1 = &Значение1 ИЛИ Таблица.Поле1 = &Значение2)
И
(Таблица.Поле2 = &Значение3 ИЛИ Таблица.Поле2 = &Значение4)
```

т.к. можно переписать при помощи В (специально переписывать не нужно, можно оставить, как есть):

ГДЕ

```
Таблица.Поле1 В (&Значения1) // Основное условие
И Таблица.Поле2 В (&Значения2) // Дополнительное условие (или наоборот)
```

3. Оператор ПОДОБНО

В основном условии для последнего из используемых или единственного поля индекса можно использовать оператор ПОДОБНО. Функции работы со строками, в некоторых случаях, можно привести к оператору ПОДОБНО и использовать его в основном условии.

НЕПРАВИЛЬНО 1:

ГДЕ

```
ПОДСТРОКА(Таблица.Поле, 1, 6) = "строка"
```

ПРАВИЛЬНО 1:

ГДЕ

```
Таблица.Поле ПОДОБНО "строка%"
```

НЕПРАВИЛЬНО 2:

ГДЕ

```
ПОДСТРОКА(Таблица.Поле, 3, 6) = "строка"
```

НЕПРАВИЛЬНО 2:

ГДЕ

```
Таблица.Поле ПОДОБНО "__строка%" // Литерал не должен начинаться с символов "__" или "%"
```

ПРАВИЛЬНО 2:

Добавить новое вычисляемое при записи в таблицу поле, которое будет содержать фрагмент ПОДСТРОКА(Таблица.Поле, 3, 6). Проиндексировать это поле и искать по следующему условию:

ГДЕ

```
Таблица.ВычисляемоеПоле ПОДОБНО "строка%"
```

4. Оператор МЕЖДУ

В основном условии для последнего из используемых или единственного поля индекса можно использовать оператор МЕЖДУ. Функции работы с датой, в некоторых случаях, можно привести к оператору МЕЖДУ и использовать его в основном условии.

НЕПРАВИЛЬНО:

ГДЕ

```
МЕСЯЦ(Таблица.Поле) = 1
```

ПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле МЕЖДУ &датаНачалаМесяца И &датаКонцаМесяца
```

Например, ДатаНачалаМесяца=01.01.2016, ДатаКонцаМесяца=31.01.2016 23:59:59

5. Выражение ВЫБОР

Выражение ВЫБОР можно использовать только в дополнительных условиях.

ПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)
И // Дополнительное условие (можно использовать ВЫБОР)
ВЫБОР
    КОГДА Таблица.Поле2 = &Значение2
        ТОГДА Таблица.Поле3 = &Значение3
    ИНАЧЕ Таблица.Поле4 = &Значение4
КОНЕЦ
```

НЕПРАВИЛЬНО:

ГДЕ

```
ВЫБОР // Основное условие (поиск по индексу использоваться не будет)
КОГДА Таблица.Поле2 = &Значение2
ТОГДА Таблица.Поле3 = &Значение3
ИНАЧЕ Таблица.Поле4 = &Значение4
КОНЕЦ
```

6. Арифметические операции

Арифметические операции над полями можно выполнять только в дополнительных условиях.

ПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле1 = &Значение1 // Основное условие (использует индекс)
И // Дополнительное условие (можно выполнять арифметические операции)
Таблица.Поле2 - 1 > 0
```

НЕПРАВИЛЬНО:

ГДЕ

```
Таблица.Поле1 - 1 > 0 // Основное условие (поиск по индексу невозможен)
```

7. Если в конфигурации описано несколько ролей с разным ограничением доступа на уровне записей (RLS), то не следует назначать одному пользователю более одной такой роли. Если один пользователь будет включен, например, в две роли с RLS - бухгалтер и кадровик, то при выполнении всех его запросов к их условиям будут добавляться условия обоих RLS с использованием логического ИЛИ. Таким образом, даже если в исходном запросе нет условия ИЛИ, оно появится там после добавления условий RLS. Такой запрос так же может выполняться неоптимально - медленно и с избыточными блокировками.

Вместо этого следует:

- Пересмотреть состав ролей таким образом, чтобы к одному объекту метаданных давала доступ только одна роль (на чтение, запись и т.п.);
- При необходимости разработки нескольких ролей, предоставляющих доступ к одному объекту метаданных, задавать в них одинаковые условия RLS. В этом случае к тексту запроса будет добавлено только одно условие, без объединения по ИЛИ;
- Либо если это допустимо с точки зрения прикладной области, создать "смешанную" роль - "бухгалтер-кадровик" и прописать ее RLS таким образом, чтобы избежать использования ИЛИ в условиях, а пользователя включить в эту одну роль.

См. также

- [Типичные причины неоптимальной работы запросов и методы оптимизации](#) (статья на ИТС)
- [Стандартные роли](#)
- [Настройка ролей и прав доступа](#)

Разрешение итогов для периодических регистров сведений

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

#std708

Методическая рекомендация (полезный совет)

1.1. Для периодических регистров сведений рекомендуется разрешить итоги, если выполнены все следующие условия:

- в регистре ожидается большой объем данных (например, оправданно для регистра с ценами номенклатуры; но не имеет смысла для регистра с курсами валют);
- в конфигурации предусмотрены частотные запросы к срезам последних на текущий момент времени и/или к срезам первых для получения актуальных данных (т.е. когда не задан период в параметрах виртуальных таблиц СрезПервых и СрезПоследних);
- при этом остальные условия для виртуальных таблиц СрезПервых и СрезПоследних задаются только на значения измерений (и разделителей, находящихся в режиме **Независимо и совместно**);
- в ограничениях доступа к данным регистра используются только измерения (и разделители, находящиеся в режиме **Независимо и совместно**).

Полный список всех условий, когда в запросах задействуются итоги регистра сведений, см. в [документации к платформе 1С:Предприятие](#).

Например, если в конфигурации предусмотрены часто выполняющиеся запросы к регистру ЦеныНоменклатуры для получения текущих цен номенклатуры:

ВЫБРАТЬ

```
Номенклатура.Артикул КАК Артикул,
ЦеныНоменклатуры.Цена КАК Цена,
```

...

ИЗ
Справочник.Номенклатура КАК Номенклатура
ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.ЦеныНоменклатуры.СрезПоследних(, ВидЦены = &ВидЦены) КАК ЦеныНоменклатуры
ПО ЦеныНоменклатуры.Номенклатура = Номенклатура.Ссылка

...

то при соблюдении всех остальных условий, перечисленных выше, установка свойства **Разрешить итоги: срез последних** существенно ускорит выполнение таких запросов, засчет того, что выборка будет выполняться напрямую из дополнительных таблиц, в которых хранятся только последние значения (для среза последних) и первые значения (для среза первых).

1.2. Кроме того, следует рассмотреть альтернативные варианты по пересмотру запросов к регистру таким образом, чтобы эти условия выполнялись.

Например, если в некоторых случаях данные в регистр ЦеныНоменклатуры записываются будущей датой, а при подборе товаров к этому регистру выполняется запрос всегда на текущую дату (дата явно задана в параметре виртуальной таблицы СрезПоследних), то итоги не будут ускорять выполнение таких запросов. Поскольку итоги строятся только для первых и последних записей регистра.

Однако если при открытии формы подбора товаров анализировать, есть ли регистраторы с будущей датой, и если их нет – выполнять другой запрос к срезу последних без установки даты, то такой запрос будет работать быстрее.

2. Во всех остальных случаях, не следует разрешать итоги для периодических регистров сведений. Прежде всего, если

- чаще всего (всегда) к виртуальным таблицам среза первых/последних регистра сведений выполняются запросы на конкретный период (например, на дату документа).
- в условиях для виртуальных таблиц СрезПервых и СрезПоследних чаще всего (всегда) используются подзапросы и соединения (обращения «через точку» к полям связанных таблиц).

Например, в этом случае:

ВЫБРАТЬ

...

ИЗ

РегистрСведений.КурсыВалют.СрезПоследних(, Валюта.Код = &КодВалютыСклада) КАК КурсыВалют

3. Не требуется предусматривать в конфигурации отдельного механизма пересчета итогов, так как актуализация таблиц итогов выполняется автоматически при каждой записи набора записей в регистре.

Исключение составляют отдельные случаи, когда актуализация итогов при записи отключалась принудительно с помощью вызова метода РегистрСведенийМенеджер.УстановитьИспользованиеИтогов(Ложь).

Эффективное обращение к виртуальной таблице «Остатки»

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

#std733

Эффективность обращения к виртуальным таблицам во многом зависит от того, как построено обращение к этой таблице. Стандарт [Обращения к виртуальным таблицам](#) описывает общие требования и рекомендации по работе с виртуальными таблицами. В этом стандарте изложены дополнительные рекомендации по повышению эффективности обращения к виртуальной таблице **Остатки** регистров накопления и бухгалтерии.

При обращении к любой виртуальной таблице платформа 1С:Предприятие генерирует запрос к СУБД, содержащий вложенный запрос. Самым эффективным вложенным запросом для чтения остатков будет чтение хранимой таблицы текущих остатков без применения группировки по измерениям. Платформа 1С:Предприятие генерирует такой запрос, если будут соблюдены все перечисленные ниже условия:

- получение остатков ведется без указания даты;
- не используется разделение итогов (необходимо учитывать при использовании такого режима может снижаться параллельность записи в регистр. См. также [Режим разделения итогов для регистров накопления, Режим разделения итогов для регистров бухгалтерии](#));
- внешний по отношению к виртуальной таблице запрос использует все измерения (в предложении ВЫБРАТЬ или в условиях соединения).

Пример.

Регистр накопления ОстаткиТовара содержит два измерения: Склад и Номенклатура, а также ресурс Количество. Необходимо запросом получить список всей номенклатуры, с указанием количества товаров на конкретном складе.

НЕПРАВИЛЬНО

```
ВЫБРАТЬ
    СпрНоменклатура.Ссылка КАК Товар,
    ЕСТЬNULL(ОстаткиТоваров.Остаток, 0 ) КАК Остаток
ИЗ
Справочник.Номенклатура КАК СпрНоменклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки(&СегодняшняяДата, Склад = &Склад) КАК ОстаткиТоваров
ПО ОстаткиТоваров.Номенклатура = СпрНоменклатура.Ссылка
```

В этом запросе:

- в условия виртуальной таблицы передана дата, поэтому будет использована не только хранимые таблицы остатков, но и таблица движений. Т.к. необходимо получить текущие остатки, то дату в запрос передавать не нужно;
- измерение Склад не используется во внешнем по отношению к виртуальной таблице запросе, поэтому вложенный запрос остатков будет содержать группировку этому измерению.

ПРАВИЛЬНО

```
ВЫБРАТЬ
    СпрНоменклатура.Ссылка КАК Товар,
    ЕСТЬNULL(ОстаткиТоваров.Остаток, 0 ) КАК Остаток
ИЗ
Справочник.Номенклатура КАК СпрНоменклатура
    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки(, Склад = &Склад) КАК ОстаткиТоваров
ПО ОстаткиТоваров.Номенклатура = СпрНоменклатура.Ссылка
    И ОстаткиТоваров.Склад = &Склад
```

Использование временных таблиц

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

#std777

1. В общем случае, временные таблицы рекомендуется использовать для повышения производительности и стабильности выполнения запросов. Их можно использовать для других целей (например, для улучшения архитектуры кода), но при этом, нужно понимать, что это может в некоторых случаях приводить к снижению производительности.

2. Есть ситуации, когда временные таблицы не следует создавать или создание временных таблиц необходимо минимизировать.

2.1. Не следует создавать временные таблицы с очень большим объемом данных (сотни тысяч записей). Иначе это приведет к существенному снижению производительности при записи и исчерпанию свободного места на диске. Если алгоритму требуется работать с большим объемом данных, то он должен выполнять обработку этих данных порциями.

2.2. Следует максимально ограничивать количество данных, выбираемых во временную таблицу. Не следует помещать во временную таблицу больше данных, чем требуется последующим запросом.

2.3. Не следует помещать во временную таблицу поля, которые не используются в последующих запросах, т.к. время и место для их размещения тратится впустую.

2.4. Не следует создавать и удалять временные таблицы в цикле, если можно создать одну временную таблицу до выполнения цикла.

2.5. Не следует копировать одну временную таблицу в другую только ради того, чтобы переименовать первую таблицу во вторую. Вместо этого, следует передавать имя таблицы.

3. Временные таблицы следует всегда индексировать, когда это даст прирост производительности.

3.1. Индекс следует строить если:

3.1.1. Большая временная таблица участвует в соединении (не важно, с какой стороны). В индекс следует добавлять поля, участвующие в условии ПО.

3.1.2. Обращение к временной таблице выполняется в подзапросе конструкции логического оператора **B (...)**. В индекс следует добавлять поля временной таблицы из списка выбора, соответствующие перечисленным с левой стороны логического оператора **B (...)**.

См. также: стандарт 652 «[Несоответствие индексов и условий запроса](#)».

3.2. Маленькие временные таблицы индексировать не нужно (менее 1000 записей).

3.3. Если условий выбора или соединений с временной таблицей больше одного, и только одно из них проверяется часто, то индекс следует строить для наиболее часто проверяемого условия.

См. также

- [Использование вложенных запросов в условии соединения](#)
- [Ограничения на соединения с вложенными запросами и виртуальными таблицами](#)

Транзакции: правила использования

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

#std783

Транзакции применяются для целостного изменения связанных данных, т.е. все действия с базой данных, выполняемые в рамках транзакции или выполняются целиком, или целиком откатываются.

1. Использование транзакций в 1С:Предприятии обладает рядом особенностей:

- не поддерживаются вложенные транзакции (см. подробнее [Вложенность транзакций](#));
- при возникновении исключения в общем случае транзакция не может быть зафиксирована – при этом не важно, было ли это исключение обработано или нет (см. подробнее [Ошибки базы данных и транзакции, Особенности работы объектов при отмене транзакции](#));
- транзакция может быть инициирована явно в прикладном коде при использовании метода **Начать Транзакцию**. Так же платформа 1С:Предприятие нейтральным образом начинает транзакцию при любой записи в базу данных (см. подробнее [Документация платформы, Механизм транзакций](#));

Эти особенности накладывают ряд требований к написанию кода с использованием транзакций. Несоблюдение этих требований может приводить к возникновению ошибок вида «В этой транзакции уже происходили ошибки», которые может быть крайне сложно воспроизвести и отладить.

1.1. Поскольку исключение не отменяет транзакцию сразу, но запрещает успешное завершение транзакции, то все вызовы **Начать Транзакцию** с одной стороны и **Зафиксировать Транзакцию** или **Отменить Транзакцию** с другой стороны должны быть парными.

1.2. Начало транзакции и ее фиксация (отмена) должны происходить в контексте одного метода

Правильно

Процедура ЗаписатьДанныеВИБ()

```
НачатьТранзакцию();  
  
Попытка  
    ... // чтение или запись данных  
    ДокументОбъект.Записать()  
    ЗафиксироватьТранзакцию();  
Исключение  
    ОтменитьТранзакцию();  
    ... // дополнительные действия по обработке исключения  
КонецПопытки;
```

КонецПроцедуры

Неправильно

Процедура ЗаписатьДанныеВИБ()

```
НачатьТранзакцию();  
ЗаписатьДокумент();
```

КонецПроцедуры;

Процедура ЗаписатьДокумент()

```
Попытка  
    ... // чтение или запись данных  
    ДокументОбъект.Записать()  
    ЗафиксироватьТранзакцию();  
Исключение  
    ОтменитьТранзакцию();  
    ... // дополнительные действия по обработке исключения  
КонецПопытки;
```

КонецПроцедуры

1.3. При использовании транзакций необходимо предусмотреть обработку исключений, придерживаясь следующих правил:

- метод **Начать Транзакцию** должен быть за пределами блока **Попытка-Исключение** непосредственно перед оператором **Попытка**;
- все действия, выполняемые после вызова метода **Начать Транзакцию**, должны находиться в одном блоке Попытка, в том числе чтение, блокировка и обработка данных;
- метод **Зафиксировать Транзакцию** должен идти последним в блоке **Попытка** перед оператором **Исключение**, чтобы гарантировать, что после **Зафиксировать Транзакцию** не возникнет исключение;
- необходимо предусмотреть обработку исключений – в блоке **Исключение** нужно сначала вызвать метод **Отменить Транзакцию**, а затем выполнять другие действия, если они требуются;
- рекомендуется в блоке **Исключение** делать запись в [журнал регистрации](#);
- при использовании вложенных транзакций (см. п. 1.4) в конце блока **Исключение** рекомендуется добавить оператор **Вызвать Исключение**. В противном случае исключение не будет передано выше по стеку вызовов, там не сработает обработка исключения, внешняя транзакция не будет явным образом отменена и платформа вызовет исключение «В данной транзакции происходила ошибка»

Пример

```
НачатьТранзакцию();  
Попытка  
    Блокировкаданных = Новый Блокировкаданных;  
    ЭлементБлокировкаданных = Блокировкаданных.Добавить("Документ.ПриходнаяНакладная");  
    ЭлементБлокировкаданных.УстановитьЗначение("Ссылка", СсылкаДляОбработки);  
    ЭлементБлокировкаданных.Режим = РежимБлокировкаданных.Исключительный;  
    Блокировкаданных.Заблокировать();  
  
    ... // чтение или запись данных  
  
    ДокументОбъект.Записать();  
  
    ЗафиксироватьТранзакцию();  
Исключение  
    ОтменитьТранзакцию();  
  
    ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),  
        УровеньЖурналаРегистрации.Ошибка,  
        '  
        ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));  
  
    ВызватьИсключение; // есть внешняя транзакция  
  
КонецПопытки;
```

1.4. Использование вложенных транзакций приводит к усложнению кода. Принимая решение об использовании этой возможности, нужно очень внимательно оценить решаемую задачу: возможно, это усложнение просто не оправдано.

1.4.1. Не стоит усложнять код, явно используя метод **НачатьТранзакцию**, когда кроме записи объекта другие действия с базой данных не делаются – платформа при записи сама откроет транзакцию.

Не нужно явно открывать транзакцию тогда, когда не требуется выполнять [ответственное чтение данных](#). Например, обычно ответственное чтение не требуется при записи нового объекта (нового набора записей регистра).

При использовании методов **ПолучитьОбъект** (или **Прочитать** для наборов записей) необходимо анализировать должно ли чтение быть ответственным и в зависимости от этого принимать решение о явном использовании метода **НачатьТранзакцию**.

Правильно

Попытка

```
ДокументОбъект = Документы.ПриходнаяНакладная.СоздатьДокумент();
... // действия по заполнению объекта
ДокументОбъект.Записать();
Иключение
... // действия по обработке исключения
КонецПопытки;
```

Неправильно

НачатьТранзакцию();

Попытка

```
ДокументОбъект = Документы.ПриходнаяНакладная.СоздатьДокумент();
... // действия по заполнению объекта
ДокументОбъект.Записать();
ЗафиксироватьТранзакцию();
Иключение
ОтменитьТранзакцию();
КонецПопытки;
```

1.4.2. Если метод рассчитан на вызов только в рамках уже открытой транзакции (например, метод предназначен для вызова только из событий **ПередЗаписью**, **ОбработкаПроведения** и т.п.) в общем случае явным образом открывать в нем транзакцию не имеет никакого практического смысла.

1.4.3. При необходимости повысить качество сообщений об ошибках – на каждом уровне разработчик может предусмотреть свою обработку исключений, для чего, возможно, потребуется открыть вложенную транзакцию.

Пример

Вызывается метод **ДобавитьЭлектроннуюПодпись**. Внутри, если что-то пошло не так, нужно обработать исключение и добавить текст вида: «Не удалось добавить электронную подпись к объекту %ПредставлениеОбъекта% по причине: %ОписаниеОшибка%.». В противном случае исключение будет обработано выше по стеку вызовов, например, при записи файла и будет выдано сообщение вида: «Не удалось записать файл %ИмяФайла% по причине: %ОписаниеОшибка%, где в «%ОписаниеОшибка%», будет просто указание на строчку кода и пользователю будет не понятно, зачем вообще программа записывала файл, если он просто его подписывал.

1.4.4. При обработке исключения, если транзакция все еще активна, например, исключение возникло во вложенной транзакции, нельзя обращаться к базе данных, так как это приведет к исключению «В этой транзакции уже происходили ошибки». При этом нужно учитывать, что обращение к базе данных может быть неявным, например, для получения представления ссылки.

2. Ограничение на длину транзакции.

2.1. В общем случае в рамках одной транзакции нужно выполнять только те действия, которые неделимы, исходя из бизнес-логики.

Пример

При проведении документа записывается документ и его движения в регистрах. Если не прошла запись хотя бы в один регистр вся операция проведения должна быть отменена.

2.1.1. Если с точки зрения бизнес-логики действия могут быть выполнены по отдельности, то их в общем случае не следует объединять в одну транзакцию.

2.1.2. Исключением из п.2.1.1 могут быть случаи, когда с целью оптимизации нескольких несвязанных объектов обрабатываются в рамках одной транзакции. В этом случае необходимозвешенно подходить к выбору порций обработки данных: нужно стремиться к достижению золотой середины между длительностью одной транзакции и объемом фиксируемых данных с одной стороны и количеством транзакций с другой.

2.2. Следует избегать транзакций, которые выполняются длительное время.

Пример

Неправильно

Для загрузки адресного классификатора ФИАС записывать все данные, относящиеся к одной версии классификатора в одной транзакции, для того, чтобы в случае ошибки откатить целиком загружаемую версию классификатора.

Т.к. данных по одной версии классификатора много (объем около 1 Гб), то для выполнения такой транзакции, во-первых, может не хватить оперативной памяти (особенно при использовании файловой информационной базы на 32-разрядной ОС), а, во-вторых, такая операция будет выполняться достаточно долго и ее нельзя будет оптимизировать за счет выполнения в несколько потоков.

Правильно

Разбить загрузку новой версии классификатора ФИАС на небольшие транзакции и реализовать функциональность по откату к предыдущей версии в случае ошибки.

См. также [Особенности использования транзакций при обмене данными](#)

2.2.1 Чем дольше выполняется транзакция, тем большее время будут заняты ресурсы сервера **1С:Предприятия** и СУБД. Как правило длинные транзакции занимают следующие ресурсы:

- в ходе выполнения транзакции все изменения в базе данных записываются в журнал транзакций, что необходимо для возможности откатить транзакцию;
- блокировки, установленные в транзакции, остаются до конца транзакции;
- на сервере **1С:Предприятия** блокировки занимают оперативную память;
- другие ресурсы, необходимые самой бизнес-логике, которая выполняется в транзакции.

Все это в целом может снижать эффективность использования ресурсов.

2.2.2. Если две транзакции пересекаются по блокируемым ресурсам, то транзакция, которая начала выполняться позже, будет ожидать возможность установления блокировки ограниченное время (по умолчанию – 20 секунд), после чего будет завершена с исключением «Превышено время ожидания установки блокировки». Поэтому длинные транзакции могут сильно снижать удобство параллельной работы пользователей.

Возникновение таких исключений – это повод провести анализ действий, которые выполняются в конфликтующих транзакциях

- возможно, какие-то действия можно вынести за транзакцию (см. п. 2.4);
- если действие вынести нельзя, то нужно постараться оптимизировать алгоритм его выполнения;
- так же нужно проанализировать оптимальность устанавливаемых блокировок (см. группу стандартов [Избыточные блокировки и методы оптимизации](#))

2.3. В рамках транзакции нужно стремиться выполнять минимум действий – только те, которые нельзя в соответствии с бизнес-логикой выполнять вне транзакции. В частности:

- сложные, ресурсоемкие расчеты нужно стремиться делать до начала транзакции, если это позволяет бизнес-логика;
- если расчет должен выполняться в транзакции, то нужно стремиться сделать его как можно более простым. Например, контроль остатков можно делать уже после записи простым запросом к записываемому регистру;
- проверка заполнения объекта должна делаться вне транзакции (см. [Проверки, выполняемые в и вне транзакции записи объекта](#));

- запросы, перед выполнением которых не нужно устанавливать блокировку данных, нужно стремиться выполнять до начала транзакции (см. [Ответственное чтение данных](#));
- запросы, выполняемые в рамках транзакций нужно стремиться оптимизировать (см. группу стандартов [Оптимизация запросов](#))

Использование управляемого режима блокировки

Область применения: управляемое приложение, обычное приложение.

#std460

В конфигурациях следует использовать "Управляемый" режим блокировок (свойство **Режим управления блокировкой данных** конфигурации устанавливается в значение **Управляемый**) и учитывать особенности работы в этом режиме, в частности:

- Чтение данных другими транзакциями будет невозможно только в том случае, если в текущей и других транзакциях устанавливаются несовместимые управляемые блокировки.
- Явная управляемая блокировка должна устанавливаться перед чтением данных:
 - есличитываются данные, которые в дальнейшем должны быть изменены;
 - необходимо обеспечить неизменность считываемых данных до конца транзакции.
- При установке управляемой блокировки необходимо стремиться, чтобы блокировка была установлена только на те записи, которые будут обработаны системой в результате отработки программного кода.
- Не следует применять в запросах конструкцию **ДЛЯ ИЗМЕНЕНИЯ**, как не имеющую смысла в этом режиме.

При работе в автоматическом режиме управления блокировкой 1С:Предприятие устанавливает высокую степень изоляции данных в транзакции на уровне СУБД. Это позволяет полностью исключить возможность получения нецелостных или некорректных данных без каких-либо специальных усилий со стороны прикладных разработчиков.

Но, при этом могут возникать избыточные блокировки на уровне СУБД. Эти блокировки связаны как с особенностями реализации механизмов блокировок в самой СУБД, так и с тем, что СУБД не может учитывать (и не учитывает) физический смысл и структуру объектов метаданных 1С:Предприятия.

См. также

- [Использование транзакций при чтении данных](#)
- [Общие сведения об избыточных блокировках](#)
- методика перевода приложений в управляемый режим блокировки содержится в статье «[Блокировки данных в 1С:Предприятии 8](#)»

Блокировка данных объекта для редактирования из кода

Область применения: управляемое приложение, обычное приложение.

#std490

1.1. Прежде чем изменять существующий объект информационной базы из кода на встроенным языке, следует предварительно его заблокировать (установить «блокировку данных для редактирования» или «объектную блокировку»), тем самым, во-первых, убедиться, не заблокирован ли он другими объектами, во-вторых, попытаться предотвратить его изменение другими пользовательскими сессиями (или другими экземплярами объекта в этом же сеансе).

В противном случае, если при изменении и записи из встроенного языка не устанавливать блокировку объекта на время редактирования, то может возникнуть, например, ситуация, когда пользователь не сможет сохранить свои изменения, если эти же самые данные были конкурентно изменены в другом сеансе.

При этом блокировка данных для редактирования не запрещает запись заблокированных данных в других пользовательских сессиях (или в других экземплярах объекта в этом же сеансе), а лишь не позволяет нескольким объектам одновременно установить блокировку одних и тех же данных. В отличие от [транзакционных блокировок данных](#), пессимистическая блокировка данных для редактирования предназначена для обеспечения конкурентной работы пользователей с объектами информационной базы 1С:Предприятие (элементами справочников, документами и т.д.) Подробнее о блокировке данных для редактирования см. документацию по платформе **1С:Предприятие 8**.

1.2. Для блокировки данных для редактирования из встроенного языка следует вызывать метод объектов **Заблокировать** или метод глобального контекста **ЗаблокироватьДанныеДляРедактирования**.

Пример № 1. Требуется заблокировать объект и, если это удалось, модифицировать данные. В противном случае – проинформировать пользователя об отказе в выполнении операции с помощью сообщения вида:

«Не удалось заблокировать запись. Действие (изменение, удаление или блокировка записи) не выполнено. Ошибка блокировки объекта. Объект уже заблокирован: компьютер: <имя компьютера>, пользователь: <имя пользователя>, сеанс: <номер сеанса>, начал: <дата и время>, приложение: <тип клиентского приложения>».

```
ФайлОбъект = ДанныеФайла.Ссылка.ПолучитьОбъект();
// Выполнить блокировку объекта от изменения другими режимами
// или пользователями; в случае блокировки -
// вывести пользователю сообщение об исключении.
ФайлОбъект.Заблокировать();
// Затем изменить и записать объект
ФайлОбъект.Редактирует = Справочники.Пользователи.ПустаяСсылка();
ФайлОбъект.Записать();
```

Аналогичным образом, можно воспользоваться методом глобального контекста **ЗаблокироватьДанныеДляРедактирования**:

```
ФайлОбъект = ДанныеФайла.Ссылка.ПолучитьОбъект();
// Выполнить блокировку объекта от изменения другими режимами
// или пользователями; в случае блокировки -
// вывести пользователю сообщение об исключении.
ЗаблокироватьДанныеДляРедактирования(ДанныеФайла.Ссылка);
// Затем изменить и записать объект
ФайлОбъект.Редактирует = Справочники.Пользователи.ПустаяСсылка();
ФайлОбъект.Записать();
```

Пример № 2. Требуется пропустить обработку объекта, если он заблокирован для редактирования. При очередном вызове процедуры (например, из фонового или регламентного задания) будет предпринята повторная попытка изменения объекта.

```
ФайлОбъект = ТекущаяВерсия.ПолучитьОбъект();
// Выполнить блокировку объекта от изменения другими режимами или пользователями.
УстановитьПолноеНаименование = Истина;
Попытка
    ФайлОбъект.Заблокировать();
Исключение
    // в случае блокировки - не выполнять изменение объекта
    УстановитьПолноеНаименование = Ложь;
    // записать предупреждение в журнал регистрации
    ЗаписьЖурналаРегистрации(НСтр("ги = 'Фоновое обновление имен файлов'", Метаданные.ОсновнойЯзык.КодЯзыка),
        УровеньЖурналаРегистрации.Предупреждение, ФайлОбъект, ПодробноеПредставлениеОшибка(ИнформацияОб ошибке()));
КонецПопытки;
```

```
// Пропустить обработку объекта, если он заблокирован.
Если УстановитьПолноеНаименование Тогда
    ФайлОбъект.ПолноеНаименование = ПолноеНаименование;
    ФайлОбъект.Записать();
КонецЕсли;
```

1.3. При редактировании данных в формах, платформа 1С:Предприятие автоматически устанавливает блокировку объекта, указанного в качестве основного реквизита формы.

2. Не следует проверять блокировку объектов для редактирования в следующих случаях:

- при выполнении отдельных операций, имеющих по логике работы больший приоритет по сравнению с интерактивными действиями пользователя. Например, загрузка данных при обмене;
- при действиях, которые гарантированно выполняются в монопольном режиме. Например, в процедурах обновления и первоначального заполнения данных информационной базы.

Ответственное чтение данных

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

#std648

1. Общие рекомендации по использованию транзакций при чтении данных

Область применения (уточнение): управляемое приложение, обычное приложение.

1.1. Если чтение данных из информационной базы должно быть ответственным, следует производить такое чтение в транзакции с предварительной установкой [управляемых блокировок](#). В общем случае, ответственный следует считать любое чтение, на основе результатов которого производятся какие-либо изменения в информационной базе или принимаются решения. Например, ответственное чтение данных требуется в следующих случаях:

- Чтение данных при проведении, для последующего формирования движений;
- Чтение данных для последующей целостной передачи в другую систему, например в программы типа «Клиент банка»;
- Выполнение групповой обработки объектов, при реструктуризации данных в обработчиках отложенного и оперативного обновления ИБ (*)

* Примечание: перед модификацией ссылочных объектов, обычно, следует устанавливать на них [пессимистичные объектные блокировки](#).

Неправильно:

```
// 1. Прочитать регистр сведений
Запрос = Новый Запрос(
    "ВЫБРАТЬ РАЗРЕШЕННЫЕ
    | ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок
    | ИЗ
    | РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету
    | ГДЕ
    | ЗаметкиПоПредмету.Предмет = &Предмет";
Запрос.УстановитьПараметр("Предмет", ПредметЗаметок);
Выборка = Запрос.Выполнить().Выбрать();

КоличествоЗаметок = 0;
Если Выборка.Следующий() Тогда
    КоличествоЗаметок = Выборка.КоличествоЗаметок;
КонецЕсли;

// 2. Записать в регистр сведений
НаборЗаписей = РегистрыСведений.ЗаметкиПоПредмету.СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Предмет.Установить(ПредметЗаметок);
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись.Предмет = ПредметЗаметок;
НоваяЗапись.КоличествоЗаметок = КоличествоЗаметок + 1;
НаборЗаписей.Записать();
```

Правильно:

```
// 1. Начать транзакцию для пакета из двух операций чтения и записи регистра
НачатьТранзакцию();
```

Попытка

```
// 2. Установить исключительную блокировку на интересующий диапазон записей регистра,
// для того чтобы гарантировать, что в момент записи количество заметок не изменилось с момента чтения в каком-либо другом
сеансе.
Блокировкаданных = Новый Блокировкаданных;
ЭлементБлокировкаданных = Блокировкаданных.Добавить("РегистрСведений.ЗаметкиПоПредмету");
ЭлементБлокировкаданных.УстановитьЗначение("Предмет", ПредметЗаметок);
ЭлементБлокировкаданных.Режим = РежимБлокировкаданных.Исключительный;
Блокировкаданных.Заблокировать();
```

// 3. Прочитать регистр сведений

```
Запрос = Новый Запрос(
    "ВЫБРАТЬ РАЗРЕШЕННЫЕ
    | ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок
    | ИЗ
    | РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету
    | ГДЕ
    | ЗаметкиПоПредмету.Предмет = &Предмет";
Запрос.УстановитьПараметр("Предмет", ПредметЗаметок);

Выборка = Запрос.Выполнить().Выбрать();
```

КоличествоЗаметок = 0;

Если Выборка.Следующий() Тогда

КоличествоЗаметок = Выборка.КоличествоЗаметок;

КонецЕсли;

// 4. Записать в регистр сведений

```
НаборЗаписей = РегистрыСведений.ЗаметкиПоПредмету.СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Предмет.Установить(ПредметЗаметок);
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись.Предмет = ПредметЗаметок;
НоваяЗапись.КоличествоЗаметок = КоличествоЗаметок + 1;
НаборЗаписей.Записать();
```

ЗафиксироватьТранзакцию();

Иключение

```
// 5. Если при установке блокировки возникла исключительная ситуация из-за того, что регистр уже заблокирован в другом сеансе
// или по другим причинам,
// отменить транзакцию и записать сведения об ошибке в журнал регистрации.
ОтменитьТранзакцию();
ЗаписьЖурналаРегистрации(НСтр("ги = 'Заметки'", ОбщегоНазначения.КодОсновногоЙзыка()), УровеньЖурналаРегистрации.Ошибка,,,
ПодробноеПредставлениеОшибки(ИнформацияОбоОшибке()));
ВызватьИключение;
КонецПопытки;
```

В некоторых случаях, ответственное чтение не требуется в силу решаемой прикладной задачи, например:

- Получение данных динамическими списками;
- Поиска данных;
- Формирование большинства отчетов.

В некоторых случаях, ответственное чтение не требуется, так как конкурентная работа с данными маловероятна или полностью исключена, например:

- Обращение к условно постоянной информации. Например, чтение константы **ВалютаРегламентированногоУчета** или обращение к учетной политике;
- Действия, которые гарантированно выполняются в монопольном режиме. Например, в процедурах обновления и первоначального заполнения данных информационной базы;
- Действия над данными, доступ к которым имеет только один пользователь, поэтому конкурентная работа с ними маловероятна или полностью исключена. Например, персональные данные, хранящиеся в «разрезе» пользователей;
- Мобильное приложение, где конкурентная работа с данными маловероятна или полностью исключена.

1.2. В большинстве случаев, при выполнении чтения в обработчиках событий связанных с модификацией данных, весь код обработчика выполняется в рамках системной транзакции, которая открыта платформой, и явно открывать новую транзакцию не требуется.

Например, в системной транзакции выполняются обработчики модулей объектов и соответствующие им подписи на события:

- [ПередЗаписью](#);
- [ПриЗаписи](#);
- [ПередУдалением](#).

Подробнее – см. [документацию к платформе 1С:Предприятие](#).

Область применения (уточнение): управляемое приложение, обычное приложение.

2. Выбор: исключительная или разделяемая блокировка

2.1. Если в транзакции производится ответственное чтение данных с их последующим изменением, необходимо установить **исключительную управляемую блокировку** (до выполнения чтения). В противном случае возможно возникновение взаимоблокировки.

Пример установки исключительной блокировки (без открытия транзакции – в предположении, что ранее уже была открыта системная транзакция):

```
// 1. Установить исключительную блокировку для ответственного чтения объекта с целью его дальнейшего изменения
Блокировка = Новый Блокировкаданных;
ЭлементБлокировки = Блокировка.Добавить("Справочник.Приказы");
ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказСсылка);
ЭлементБлокировки.Режим = РежимБлокировкиданных.Исключительный; // можно не указывать, т.к. по умолчанию Исключительный
Блокировка.Заблокировать();
```

```
// 2. Получить объект для его дальнейшей модификации
Объект = ПриказСсылка.ПолучитьОбъект();
Если Объект = Неопределено Тогда // объект может быть уже удален в других сессиях
    Возврат;
КонецЕсли;
```

```
// 3. Выполнить блокировку объекта от изменения другими сессиями
ЗаблокироватьДанныеДляРедактирования(ПриказСсылка);
```

```
// 4. Записать измененный объект
Объект.Реквизит = ...
Объект.Записать();
```

2.2. Если в транзакции производится ответственное чтение данных без их последующего изменения (например, для формирования движений), необходимо установить разделяемую блокировку на читаемые данные и исключительную блокировку на изменяемые данные.

Пример установки разделяемой блокировки (без открытия транзакции – в предположении, что ранее уже была открыта системная транзакция):

```
// 1. Установить разделяемую блокировку для ответственного чтения нескольких связанных объектов
Блокировка = Новый Блокировкаданных;
ЭлементБлокировки = Блокировка.Добавить("Справочник.Приказы");
ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказСсылка);
ЭлементБлокировки.Режим = РежимБлокировкиданных.Разделяемый;
Блокировка.Заблокировать();
```

```
// 2. Прочитать первый объект - приказ
ПриказОбъект = ПриказСсылка.ПолучитьОбъект();

// 3. Прочитать второй объект - пользователя (автора приказа)
Блокировка = Новый Блокировкаданных;
ЭлементБлокировки = Блокировка.Добавить("Справочник.Пользователи");
ЭлементБлокировки.УстановитьЗначение("Ссылка", ПриказОбъект.Автор);
ЭлементБлокировки.Режим = РежимБлокировкиданных.Разделяемый;
Блокировка.Заблокировать();
```

```
АвторПриказа = ПриказОбъект.Автор.ПолучитьОбъект();
```

См. также

- [Общие сведения об избыточных блокировках](#)
- [Транзакции: правила использования](#)

Чтение отдельных реквизитов объекта из базы данных

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

#std496

При чтении отдельных реквизитов объекта из базы данных следует иметь в виду, что вызов метода **ПолучитьОбъект** или обращение к реквизитам объекта через точку от ссылки приводят к загрузке объекта из базы целиком, вместе с его табличными частями.

Поэтому для чтения значений отдельных реквизитов из базы данных следует использовать запрос. Например, неправильно:

Процедура ЗаполнитьКодИНаименованиеСтраны()

```
СтранаСсылка = ... // получаем ссылку на элемент справочника
КодСтраны = СтранаСсылка.Код; // первое обращение загружает объект целиком
НаименованиеСтраны = СтранаСсылка.Наименование;
```

КонецПроцедуры

Правильно:

```
Процедура ЗаполнитьКодИНаименованиеСтраны()
```

```
Запрос = Новый Запрос()
"ВЫБРАТЬ"
| СтраныМира.Код,
| СтраныМира.Наименование
|ИЗ
| Справочник.СтраныМира КАК СтраныМира
|ГДЕ
| СтраныМира.Ссылка = &Ссылка";
Запрос.УстановитьПараметр("Ссылка", Ссылка);
```

```
Выборка = Запрос.Выполнить().Выбрать();
Выборка.Следующий();
```

```
КодСтраны = Выборка.Код;
НаименованиеСтраны = Выборка.Наименование;
```

КонецПроцедуры

Для упрощения синтаксиса рекомендуется также использовать специальные функции **ЗначенияРеквизитовОбъекта** или **ЗначениеРеквизитаОбъекта** (входят в состав [Библиотеки стандартных подсистем](#)).

В этом случае исходный пример будет выглядеть так:

```
Процедура ЗаполнитьКодИНаименованиеСтраны()
```

```
ЗначенияРеквизитов = ОбщегоНазначения.ЗначенияРеквизитовОбъекта(СтранаСсылка, "Код, Наименование");
КодСтраны = ЗначенияРеквизитов.Код;
НаименованиеСтраны = ЗначенияРеквизитов.Наименование;
```

КонецПроцедуры

Запись событий в историю работы пользователя

Область применения: управляемое приложение.

#std497

В историю работы автоматически попадают события интерактивного добавления или изменения объектов информационной базы (документов, элементов справочника и пр.) Дополнительно рекомендуется записывать в историю работы пользователя и другие события, которые приводят к записи объектов в результате действий пользователя. Например: команда «Поместить файл» для элемента справочника **Файлы**.

Для добавления событий в историю работы пользователя предназначен объект **ИсторияРаботыПользователя** типа **МенеджерИсторииРаботыПользователя**. Пример:

```
&НаКлиенте
Процедура ПоместитьФайл(Команда)
// Поместить файл в базу
//
// И добавить событие в историю работы пользователя
ИсторияРаботыПользователя.Добавить(ПолучитьНавигационнуюСсылку(Объект.Ссылка));
КонецПроцедуры
```

Общие сведения об избыточных блокировках

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

#std659

Методическая рекомендация (полезный совет)

1. Проектные ошибки при выборе и проектировании структуры того или иного объекта метаданных для реализации прикладной функциональности, могут привести к большому количеству избыточных блокировок и как следствие к серьезному падению общей производительности системы.

Ожидание на блокировке данных происходит в том случае, если две различные сессии **1С:Предприятия** пытаются захватить один и тот же ресурс. При работе с разными ресурсами ожидание на блокировке не происходит. В данном контексте термин «ресурс» используется в качестве обозначения неделимой совокупности данных, которая блокируется (или не блокируется) только вся целиком.

Таким образом, вопрос сводится к тому, какие именно ресурсы захватываются при выполнении того или иного действия с данными. Или, иначе говоря, насколько «мелко нарезаны» данные **1С:Предприятия**.

При анализе структуры метаданных следует обратить внимание на следующие объекты:

- [Последовательность](#)
- [Регистры бухгалтерии](#)
- [Регистры накопления](#)

См. также

- [Блокирующее чтение остатков в начале транзакции](#)
- [Использование управляемого режима блокировки](#)
- [Несоответствие индексов и условий запроса](#)
- [Разыменование ссылочных полей составного типа в языке запросов](#)
- [Запросы, выполняющие соединение с вложенными запросами или виртуальными таблицами](#)
- [Использование вложенных запросов в условии соединения](#)
- [Фильтрация виртуальных таблиц без использования параметров](#)
- [Ограничение на использование логического ИЛИ в условиях запросов](#)

Сдвиг границы последовательности документов

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

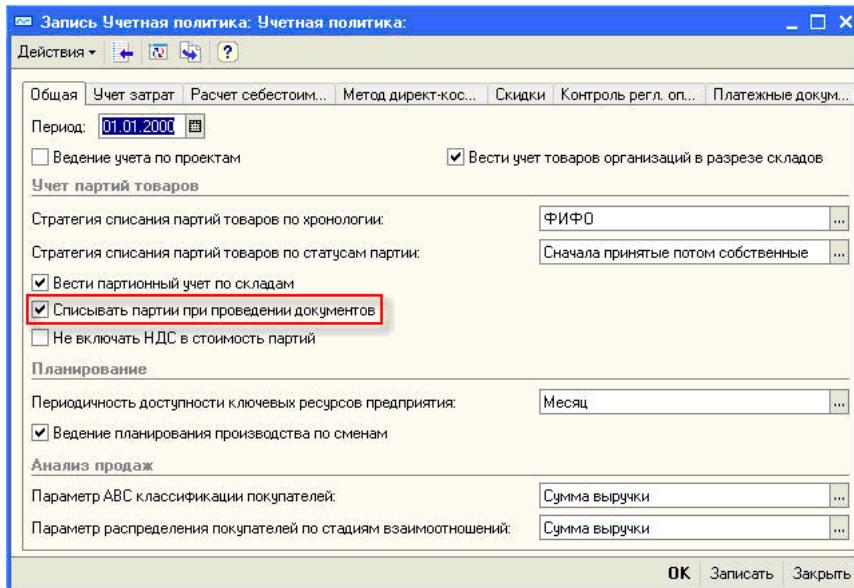
#std662

Не рекомендуется двигать границу последовательности при проведении документов. Это может привести к возникновению ожиданий на блокировках и снижению общей производительности системы. Операцию движения границы последовательности следует вынести из оперативных операций в регламентные, например выполнять регламентной обработкой с заданной частотой.

Граница последовательности по одному набору значений измерений является одним ресурсом. Это означает, что при движении границы последовательности по одному набору значений измерений разные пользователи будут пытаться захватить один и тот же ресурс, то есть будут блокировать друг друга.

Пример

В системе, построенной на базе УПП версий 1.x, используется учетная политика, предполагающая вычисление себестоимости списываемых товаров в оперативном режиме (непосредственно в момент списания).



Алгоритмы партионного учета в УПП используют последовательность «Партионный учет», имеющую одно измерение: **Организация**.

Для вычисления себестоимости списываемых товаров при проведении расходного документа необходимо переместить границу последовательности для данной организации на момент времени проводимого документа.

Если два пользователя будут одновременно проводить расходные документы по одной организации (что весьма вероятно), то они будут блокировать друг друга. Такое поведение системы не является особенностью реализации последовательностей в **1С:Предприятии**, но продиктовано требованиями самого алгоритма – необходимо знать точную последовательность расположения документов. То есть все конкурирующие по времени (одновременно проводящиеся) документы должны выстроиться друг за другом.

Движение границы последовательности при оперативном проведении документов способно значительно снизить общую производительность системы. В данном случае правильным решением было бы выключение этого флагка, то есть отказ от оперативного расчета себестоимости при проведении документов. Вместо этого следует использовать регламентную обработку, входящую в состав УПП, которая будет вычислять себестоимость с некоторой заданной частотой.

См. также

- «[Общие сведения об избыточных блокировках](#)»

Режим разделения итогов для регистров бухгалтерии

Область применения: управляемое приложение, обычное приложение.

#std663

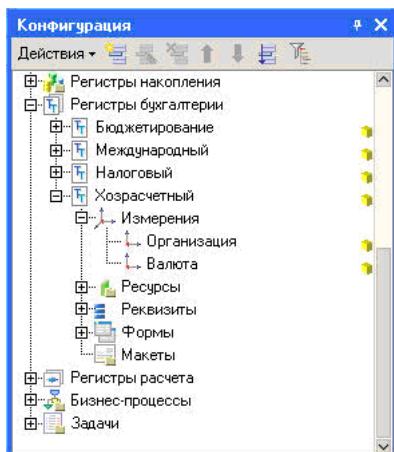
1. Если в системе осуществляется оперативная запись движений по бухгалтерскому регистру в многопользовательском режиме, то рекомендуется включить для данного регистра режим разделения итогов. При включенном режиме разделения итогов пользователи смогут параллельно обновлять таблицу остатков даже в том случае, если у них совпадает период, счет и значения измерений.

В противном случае таблица остатков регистра бухгалтерии может стать узким местом при конкурентной работе большого количества пользователей.

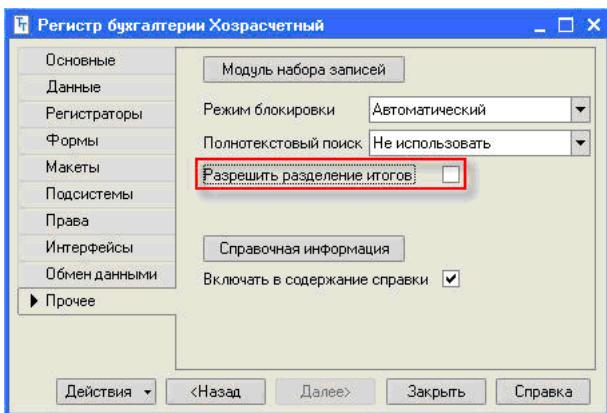
См. также [Блокирующее чтение остатков в начале транзакции](#)

Пример 1

В конфигурации определен регистр **Хозрасчетный** с измерениями **Организация** и **Валюта**.



При этом запрещено разделяние итогов регистра:



Предположим, что два пользователя одновременно проводят документы, которые осуществляют движение по данному регистру. Пользователи будут блокировать друг друга в том случае, если движения:

- относятся к одному и тому же периоду;
- относятся к одному и тому же счету;
- имеют одинаковые значения измерений, то есть организацию и валюту.

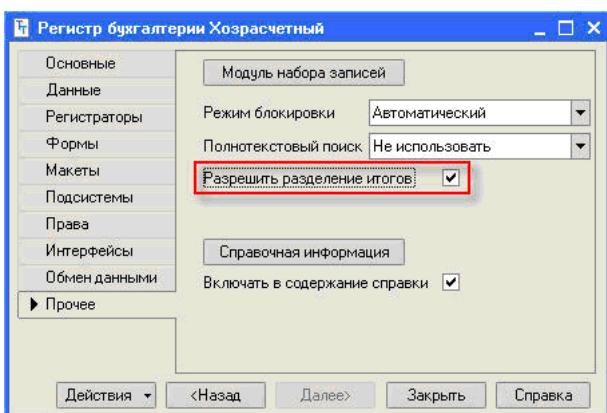
В реальной жизни одновременное выполнение перечисленных условий является весьма вероятным, поскольку большинство пользователей будет работать в одном периоде, с одним счетом и с одинаковыми значениями измерений (**организация** и **валюта**). Это может привести к возникновению ожиданий на блокировках и снижению общей производительности системы.

Для решения этой проблемы следует включить режим разделения итогов (см. следующий пример).

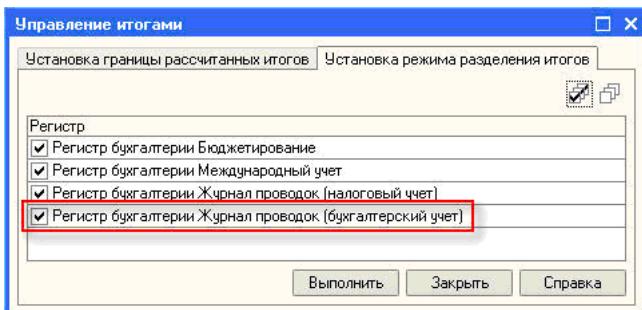
Пример 2

Если для этого же регистра разрешить и включить режим разделения итогов, то ситуация изменится.

Разрешим режим разделения итогов (в режиме **конфигурирования**):



Включим режим разделения итогов (в режиме **1С:Предприятия**):



После этого конкурирующие пользователи смогут параллельно записывать движения по регистру даже в том случае, если совпадают период, номер счета и значения всех измерений. Однако, если при этом осуществляется [контроль остатков по данному регистру](#), то эффекта от включения режима разделения не будет.

См. также

- [Устройство и использование режима разделения итогов регистров](#) (статья на ИТС)
- [Эффективное обращение к виртуальной таблице «Остатки»](#)

Режим разделения итогов для регистров накопления

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

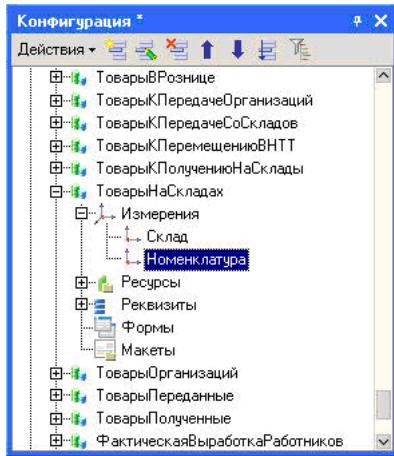
#std664

1. При проектировании регистра накопления следует помнить, что остатки по одному набору измерений хранятся в одном ресурсе регистра. То есть, степень параллельности при работе с регистром фактически зависит от состава его измерений. Состав измерений необходимо подбирать в соответствии с тем, насколько мелко должны быть «нарезаны» остатки, исходя из прикладной функциональности системы.

2. В том случае, если состав измерений не позволяет обеспечить необходимую параллельность при работе с регистром, рекомендуется использовать режим разделения итогов аналогично [регистрам бухгалтерии](#).

Следует учитывать, что режим разделения итогов не обеспечит параллельность при контроле остатков по регистру. Если контроль остатков необходим, то следует перенести его [ближе к концу транзакции](#).

Рассмотрим в качестве примера регистр накопления **ТоварыНаСкладах** со следующим составом измерений:



Пример 1

Предположим два пользователя одновременно проводят документы, которые записывают движения в данный регистр накопления. При этом первый пользователь пишет следующий набор записей:

№	Склад	Номенклатура
1	Основной склад	Кресло-качалка
2	Основной склад	Кухонный гарнитур "Тинга-2"
3	Склад №2	Мебельный гарнитур "Торэ"

Второй пользователь пишет в этот же регистр следующий набор записей:

№	Склад	Номенклатура
1	Основной склад	Мебельный гарнитур "Торэ"
2	Склад №2	Кресло-качалка

Эти наборы записей не содержат строк, совпадающих по значениям всех измерений, поэтому ожидание на блокировке в данном случае не возникнет.

Пример 2

Предположим, что второй пользователь записывает следующий набор записей

№	Склад	Номенклатура
1	Основной склад	Кухонный гарнитур "Тинга-2"
2	Склад №2	Кресло-качалка
3	Оптовый склад	Спальный гарнитур "Инга-М"

Этот набор записей содержит строку (№1), совпадающую по значениям всех измерений со строкой набора записей первого пользователя (№2).

В этом случае возникнет ожидание на блокировке и один из пользователей будет дожидаться окончания операции другого пользователя, то есть общая производительность системы снизится.

Регистр накопления также поддерживает режим разделения итогов, который позволит в данной ситуации избежать блокировки (см. Пример 3).

Пример 3

Если для данного регистра накопления включен режим разделения итогов, то будет возможна параллельная запись двух наборов, даже в том случае если они содержат одинаковые (по значениям измерений) строки.

Однако, если при этом используется контроль остатков, то режим разделения итогов не даст положительного эффекта. Иначе говоря, режим разделения итогов для регистра накопления работает аналогично [регистру бухгалтерии](#). Для снижения влияния этой блокировки на общую производительность системы [нужно перенести контроль остатков как можно ближе к концу транзакции](#).

См. также

- [Устройство и использование режима разделения итогов регистров](#) (статья на ИТС)
- [Эффективное обращение к виртуальной таблице «Остатки»](#)

Блокирующее чтение остатков в начале транзакции

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

#std661

1.1. В ряде случаев необходимо выполнять блокирующее чтение итогов. Примером такой задачи является контроль остатков при проведении документа. Если в результате проведения документа остатки станут отрицательными, то транзакция должна быть отменена (проводить такой документ нельзя).

1.2. Операция чтения остатков должна быть блокирующей, то есть необходимо запретить двум пользователям одновременно читать один и тот же остаток за период, счет и значение измерения.

Если чтение будет неблокирующим, то возможна ситуация, при которой два пользователя одновременно прочитают один и тот же остаток (например 10 единиц) и примут решение о возможности списания части этого остатка. Если сумма списаний двух пользователей будет больше 10, то в итоге остаток получится отрицательным.

Например, первый пользователь спишет 8 единиц (8 меньше 10, следовательно операция разрешена), а второй пользователь спишет 6 единиц (на таком же основании). Результатом будет -4 единицы остатка, что недопустимо с точки зрения прикладной логики системы.

2. Обычно, для контроля остатков используется запрос в модуле набора записей регистра, который идет перед записью набора. При этом возможны следующие проблемы:

- Разработчик, как правило, не контролирует порядок записи движений в разные регистры - запись обычно осуществляется автоматически платформой **1С:Предприятия**. Запрос контроля остатков реализуется в модуле набора записей и вызывается при записи движений регистра. Если этот регистр будет записываться в начале транзакции (например, первым), то установленная блокировка будет мешать работе других пользователей в течение длительного периода времени (пока будут записываться все остальные регистры), и ее влияние на производительность системы может оказаться неоправданно большим.

- В некоторых случаях, возможно, нет необходимости в контроле остатков, поскольку записываемые движения заведомо не могут привести к получению отрицательных остатков.

Для того чтобы минимизировать влияние блокирующего чтения остатков на производительность системы, необходимо:

- Проанализировать, какие именно остатки нуждаются в блокирующем чтении и при каких обстоятельствах. Например, контроль остатков не требуется при проведении приходного документа, поскольку он может только увеличить остатки. Так же не требуется контролировать остатки при перепроведении документа, который списывает в этот раз не больше остатков чем при первом проведении (этот контроль уже проводился). И так далее.
- В начале транзакции (например, в обработчике **ОбработкаПроведения** документа) в явном виде записать движения по всем регистрам, которые в данном случае не требуют контроля остатков. Следует всегда придерживаться одинакового порядка записи регистров (например, алфавитного). Необходимо обратить внимание на то, что у всех записываемых регистров накопления и бухгалтерии должен быть включен разделитель итогов, а у наборов записей свойство **БлокироватьДляИзменения** должна быть установлена в значение Ложь.
- Выполнить все остальные действия, которые должны быть выполнены в рамках этой транзакции.
- Самом конце транзакции в явном виде записать движения по тем регистрам, которые требуют контроля остатков. Для наборов записей этих регистров следует установить опцию **БлокироватьДляИзменения** в значение Истина. Это необходимо для предотвращения взаимоблокировки. В этом случае при записи набора записей будет установлена блокировка остатков регистра по данному набору значений измерений.
- Для каждого регистра выполнить запрос контроля остатков. Следует обратить внимание, что в данном случае нет необходимости использовать явную управляемую блокировку (опцию **ДЛЯ ИЗМЕНЕНИЯ** - в автоматическом режиме), поскольку проверяемые остатки уже заблокированы их записью на предыдущем шаге. Запрос должен считывать только отрицательные остатки по заданному набору значений измерений. Если такие записи имеются, то транзакция должна быть отменена. Если запрос вернул пустой результат, то транзакция должна быть зафиксирована.

Пример

В процедуре **ПередЗаписью** модуля набора записей регистра бухгалтерии **Хозрасчетный** выполняется следующий запрос:

```
Запрос.Текст = "ВЫБРАТЬ
| СуммаОстаток,
| СуммаОстатокДТ,
| СуммаОстатокКТ
| из
| РегистрБухгалтерии.Хозрасчетный.Остатки(&Период, &Счет, , Организация = &Организация)";
```

При выполнении этого запроса будут прочитаны (и заблокированы от записи) остатки по указанному условию для всех пользовательских подключений. То есть, разные ресурсы (созданные режимом разделения итогов) будут как бы объединены в один. По этой причине параллельность останется такой же, как если бы режим разделения итогов не был включен.

Для того чтобы минимизировать влияние этой блокировки на общую производительность системы, рекомендуется перенести ее как можно ближе к концу транзакции. Например, можно вынести эту проверку в модуль документа в обработчик события **ПриПроведении** после записи (в явном виде) всех движений по всем регистрам.

См. также

- [Использование транзакций при чтении данных](#)
- [Регистры бухгалтерии](#)

Тексты модулей

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

#std456

1. Тексты модулей должны быть написаны на русском языке.

Искключение составляют [веб-сервисы](#), имена методов и параметров которых рекомендуется задавать на английском языке (например, метод **TestConnection** веб-сервиса **EnterpriseDataExchange**), а также различные идентификаторы сторонних информационных систем, например, названия полей «внешних» структур данных, которые программно обрабатываются в коде.

1.1. В текстах модулях не допускается использовать букву "ё".

Исключения составляют интерфейсные тексты, выводимые пользователю в сообщениях, формах и справке, где употребление буквы «ё» в ряде случаев допустимо. Подробнее см. [Тексты](#).

1.2. Кроме того, в текстах модулей не допускается использовать неразрывные пробелы и знак минус "-" в других кодировках (короткое, длинное тире и т.п.).

Такие символы часто оказываются в тексте модулей при копировании из офисных документов и приводят к ряду сложностей при разработке. Например, не работает поиск фрагментов текста, включающих «неправильные» минусы и пробель; некорректно выводятся подсказки типов параметров процедур и функций в конфигураторе и расширенная проверка в 1С:EDT; указание «неправильного» минуса в выражениях приведет к синтаксической ошибке.

Методическая рекомендация (полезный совет)

2. Программные модули не должны иметь неиспользуемых процедур и функций.

3. Программные модули не должны иметь закомментированных фрагментов кода, а также фрагментов, которые каким-либо образом связаны с процессом разработки (отладочный код, служебные отметки, например, TODO, MRG и т.п.) и с конкретными разработчиками этого кода. Например, недопустимо оставлять подобные фрагменты в коде после завершения отладки или рефакторинга:

```
Процедура ПередУдалением(Отказ)
//   Если Истина Тогда
//     Сообщение("Для отладки");
//   КонецЕсли;
КонецПроцедуры
```

также неправильно:

```
Процедура ПередУдалением(Отказ)
  Если Истина Тогда
    // Иванов: доделать
    КонецЕсли;
КонецПроцедуры
```

Правильно: после завершения отладки или рефакторинга удалить обработчик **ПередУдалением** из кода.

4. Тексты модулей оформляются по принципу "один оператор в одной строке". Наличие нескольких операторов допускается только для "однотипных" операторов присваивания, например:

```
НачальныйИндекс = 0; Индекс = 0; Результат = 0;
```

5. Текст модуля должен быть оформлен синтаксическим отступом. Для синтаксического отступа используется табуляция (а не пробелы, чтобы при смене числа знаков в табуляции выравнивание текста сохранялось).

Размер табуляции - стандартный (4 символа).

5.1. С крайней левой позиции должны начинаться только:

- операторы **Процедура**, **КонецПроцедуры**, **Функция**, **КонецФункции**;
- операторы предварительного объявления процедур и функций;
- заголовки (описания) процедур и функций;
- объявление переменных модуля;
- операторы "раздела основной программы" (с учетом синтаксического отступа);
- директивы компилятора **&НаКлиенте**, **&НаСервере** и т.д.

- инструкции препроцессора (в т.ч. #Область и #КонецОбласти)

5.2. Процедуры **НачатьТранзакцию** и **ЗафиксироватьТранзакцию** не являются операторными скобками, поэтому текст внутри этих процедур не сдвигается.

6. При длине строки более 120 символов следует использовать переносы. Строки длиннее 120 символов делать не рекомендуется, за исключением тех случаев, когда перенос невозможен (например, в коде определена длинная строковая константа, которая выводится без переносов в окно сообщений с помощью объекта **СообщениеПользователю**).

См. также: [Перенос выражений](#).

7.1. Тексты модулей могут содержать **комментарии**. Комментарии должны быть достаточно понятными, чтобы пояснить работу модуля или комментируемого оператора. Тексты комментариев должны составляться по правилам русского языка, в деловом стиле, быть эмоционально сдержанными и не содержать слов, не относящихся к функциональности программы.

7.2. Небольшие комментарии пишутся в конце строки, которую комментируют, например:

НайденныеОшибки.Колонки.Добавить("Номер"); // для совместимости

7.3. Большие комментарии или комментарии к фрагменту кода пишутся перед комментируемым кодом в отдельной строке. Текст выравнивается по левой границе комментируемого фрагмента. Между символами комментария "://" и текстом комментария должен быть пробел.

```
// Инициализируем переменные для выполнения расчетов,
// которые выполняются далее по тексту модуля.
Текущаядата = ОбщегоНазначения.ПолучитьРабочуюДату();
ТекущийГод = Год(Текущаядата);
ТекущийМесяц = Месяц(Текущаядата);
ТекущаяНеделя = НеделяГода(Текущаядата);
ТекущийДень = День(Текущаядата);
```

8. Тексты больших процедур и функций можно разбивать на отдельные сворачиваемые области. При этом имена областей должны удовлетворять требованиям стандарта [Правила образования имен переменных](#)

Для автоматического форматирования кода можно воспользоваться обработкой [автоформатирования кода и локализации](#).

См. также

- [Правила создания общих модулей](#)
- [Структура модуля](#)

Структура модуля

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

#std455

1.1. В программном модуле (общие модули, модули объектов, модули менеджеров объектов, модули форм, команд и т.п.) в общем случае могут присутствовать следующие разделы в приведенной ниже последовательности:

- заголовок модуля
- раздел описания переменных
- экспортные процедуры и функции модуля, составляющие его программный интерфейс
- обработчики событий объекта (формы)
- служебные процедуры и функции модуля
- раздел инициализации

Некоторые разделы могут присутствовать только в модулях определенного вида. Например, обработчики событий элементов форм могут присутствовать только в модулях форм, а раздел описания переменных и раздел инициализации не могут быть определены в неглобальных общих модулях, модулях менеджеров объектов, наборов записей, значений констант и модуле сеанса.

Требование о разделении кода модуля на разделы призвано повысить читаемость кода и упростить внесение изменений в код разными авторами (разработчиками) как при коллективной разработке, так и при доработке прикладных решений на конкретных внедрениях.

1.2. Объемные разделы модулей рекомендуется разбивать на подразделы по функциональному признаку.

1.3. Разделы и подразделы оформляются в виде областей. При этом имена областей должны удовлетворять требованиям стандарта [Правила образования имен переменных](#)

1.4. Шаблон (заготовка для копирования) разделов для общих модулей:

русс.

```
#Область ПрограммныйИнтерфейс
// Код процедур и функций
#КонецОбласти
```

```
#Область СлужебныйПрограммныйИнтерфейс
// Код процедур и функций
#КонецОбласти
```

```
#Область СлужебныеПроцедурыИФункции
// Код процедур и функций
#КонецОбласти
```

англ.

```
#Region Public
// Enter code here.
#EndRegion
```

```
#Region Internal
// Enter code here.
#EndRegion
```

```
#Region Private
// Enter code here.
#EndRegion
```

- Раздел «**Программный интерфейс**» содержит экспортные процедуры и функции, предназначенные для использования другими объектами конфигурации или другими программами (например, через внешнее соединение).
- Раздел «**Служебный программный интерфейс**» предназначен для модулей, которые являются частью некоторой функциональной подсистемы. В нем должны быть размещены экспортные процедуры и функции, которые допустимо вызывать только из других функциональных подсистем этой же библиотеки.
- Раздел «**Служебные процедуры и функции**» содержит процедуры и функции, составляющие внутреннюю реализацию общего модуля. В тех случаях, когда общий модуль является частью некоторой [функциональной подсистемы](#), включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

Для объемных общих модулей рекомендуется разбивать этот раздел на подразделы, по функциональному признаку. Например:

русс.

```
#Область ОбновлениеИнформационнойБазы
// Код процедур и функций
#КонецОбласти
```

англ.

```
#Region InfobaseUpdate
// Enter code here.
#EndRegion
```

1.5. Шаблон оформления разделов для модулей объектов, менеджеров, наборов записей, обработок, отчетов и т.п.:

русс.

```
#Область ОписаниеПеременных
```

```
#КонецОбласти
```

```
#Область ПрограммныйИнтерфейс
// Код процедур и функций
#КонецОбласти
```

```
#Область ОбработчикиСобытий
// Код процедур и функций
#КонецОбласти
```

```
#Область СлужебныйПрограммныйИнтерфейс
// Код процедур и функций
#КонецОбласти
```

```
#Область СлужебныеПроцедурыИФункции
// Код процедур и функций
#КонецОбласти
```

```
#Область Инициализация
```

```
#КонецОбласти
```

англ.

```
#Region Variables
```

```
#EndRegion
```

```
#Region Public
// Enter code here.
#EndRegion
```

```
#Region EventHandlers
// Enter code here.
#EndRegion
```

```
#Region Internal
// Enter code here.
#EndRegion
```

```
#Region Private
// Enter code here.
#EndRegion
```

```
#Region Initialize
```

```
#EndRegion
```

- Раздел «**Программный интерфейс**» содержит экспортные процедуры и функции, предназначенные для использования в других модулях конфигурации или другими программами (например, через внешнее соединение). Не следует в этот раздел помещать экспортные функции и процедуры, которые предназначены для вызова исключительно из модулей самого объекта, его форм и команд. Например, процедуры заполнения табличной части документа, которые вызываются из обработки заполнения в модуле объекта и из формы документа в обработчике команды формы не являются программным интерфейсом модуля объекта, т.к. вызываются только в самом модуле и из форм этого же объекта. Их следует размещать в разделе «Служебные процедуры и функции».
- Раздел «**Обработчики событий**» содержит обработчики событий модуля объекта (**ПриЗаписи**, **ПриПроведении** и др.)
- Раздел «**Служебный программный интерфейс**» имеет такое же предназначение, как и в общих модулях.
- Раздел «**Служебные процедуры и функции**» имеет такое же предназначение, как и в общих модулях.

1.6. Шаблон оформления разделов для модулей форм:

русс.

```
#Область ОписаниеПеременных
```

```
#КонецОбласти
```

```
#Область ОбработчикиСобытийФормы
// Код процедур и функций
#КонецОбласти
```

```
#Область ОбработчикиСобытийЭлементовШапкиФормы
// Код процедур и функций
#КонецОбласти
```

```
#Область ОбработчикиСобытийЭлементовТаблицыФормы<ИмяТаблицыФормы>
// Код процедур и функций
#КонецОбласти
```

```
#Область ОбработчикиКомандФормы
// Код процедур и функций
#КонецОбласти
```

```
#Область СлужебныеПроцедурыИФункции
// Код процедур и функций
#КонецОбласти
```

англ.

```
#Region Variables
```

```
#EndRegion
```

```

#Region FormEventHandlers
// Enter code here.
#EndRegion

#Region FormHeaderItemsEventHandlers
// Enter code here.
#EndRegion

#Region FormTableItemsEventHandlers
// Enter code here.
#EndRegion

#Region FormCommandsEventHandlers
// Enter code here.
#EndRegion

#Region Private
// Enter code here.
#EndRegion

```

- Раздел «**Обработчики событий формы**» содержит процедуры-обработчики событий формы: **ПриСозданииНаСервере**, **ПриОткрытии** и т.п.
- Раздел «**Обработчики событий элементов шапки формы**» содержит процедуры-обработчики элементов, расположенных в основной части формы (все, что не связано с таблицами на форме).
- В разделах «**Обработчики событий элементов таблицы формы <имя таблицы формы>**» размещаются процедуры-обработчики таблиц формы и элементов таблиц. Для процедур-обработчиков каждой таблицы должен быть создан свой раздел.
- Раздел «**Обработчики команд формы**» содержит процедуры-обработчики команд формы (имена которых задаются в свойстве **Действие команд формы**).
- Раздел «**Служебные процедуры и функции**» имеет такое же предназначение, что и в общих модулях.

См. также: [Правила создания модулей форм](#)

1.7. Шаблон оформления разделов для модулей команд:

русс.

```

#Область ОбработчикиСобытий
// Код процедур и функций
#КонецОбласти

#Область СлужебныеПроцедурыИФункции
// Код процедур и функций
#КонецОбласти

```

англ.

```

#Region EventHandlers
// Enter code here.
#EndRegion

#Region Private
// Enter code here.
#EndRegion

```

- Раздел «**Обработчики событий**» содержит процедуру-обработчик команды **ОбработкаКоманды**.
- Раздел «**Служебные процедуры и функции**» имеет такое же предназначение, что и в общих модулях.

1.8. В модуле не должно быть пустых областей.

2. Общие требования к разделам программных модулей.

2.1. **Заголовок модуля** представляет собой комментарий в самом начале модуля. В заголовке модуля приводится его краткое описание и условия применения. Например:

```

///////////////////////////////
// Клиентские процедуры и функции общего назначения:
// - для работы со списками в формах;
// - для работы с журналом регистрации;
// - для обработки действий пользователя в процессе редактирования
// многострочного текста, например комментария в документах;
// - прочее.
/////////////////////////////

```

Для модулей форм в заголовке рекомендуется размещать описание параметров формы.

2.2. **Раздел описания переменных**. Имена переменных назначаются согласно [общим правилам образования имен переменных](#), а их использование описывается в статье [Использование глобальных переменных в программных модулях](#).

Все переменные модуля должны быть снабжены комментарием, достаточным для понимания их назначения. Комментарий рекомендуется размещать в той же строке, где объявляется переменная.

Пример:

русс.

```

#Область ОписаниеПеременных

Перем ВалютаУчета;
Перем АдресПоддержки;
...

```

#КонецОбласти

англ.

```

#Region Variables

Var PresentationCurrency;
Var SupportEmail;
...

#EndRegion

```

2.3. **Программный интерфейс**. Экспортные процедуры и функции, составляющие программный интерфейс модуля, размещаются сразу же после описания переменных. Такие процедуры и функции предназначены для использования другими объектами конфигурации или другими программами (например, через внешнее соединение), поэтому должны быть расположены в модуле на "видном месте".

См. также: [Описание процедур и функций](#)

2.4.1. **Обработчики событий формы, команд и элементов формы.** Перед служебными процедурами и функциями в модуле формы располагаются обработчики событий формы, а также обработчики событий команд и элементов формы.

Методическая рекомендация (полезный совет)

2.4.2. Рекомендуется обработчики одного элемента формы располагать вместе, придерживаясь, при этом, порядка их следования в панели свойств редактора формы в конфигураторе.

2.4.3. У каждого события должна быть назначена своя процедура-обработчик. Если одинаковые действия должны выполняться при возникновении событий в разных элементах формы следует:

- создать отдельную процедуру (функцию), выполняющую необходимые действия
- для каждого элемента формы создать отдельный обработчик с именем, назначаемым по умолчанию
- из каждого обработчика вызвать требуемую процедуру (функцию).

Например, неправильно:

```
&НаКлиенте
Процедура ПоИсполнителюПриИзменении(Элемент)
    ПараметрыОтбора = Новый Соответствие();
    ПараметрыОтбора.Вставить("ПоАвтору", ПоАвтору);
    ПараметрыОтбора.Вставить("ПоИсполнителю", ПоИсполнителю);
    УстановитьОтборСписка(Список, ПараметрыОтбора);
КонецПроцедуры
```

```
&НаКлиенте
Процедура ПоАвторуПриИзменении(Элемент)
    ПоИсполнителюПриИзменении(Неопределено);
КонецПроцедуры
```

правильно:

```
&НаКлиенте
Процедура ПоИсполнителюПриИзменении(Элемент)
    УстановитьОтбор();
КонецПроцедуры
```

```
&НаКлиенте
Процедура ПоАвторуПриИзменении(Элемент)
    УстановитьОтбор();
КонецПроцедуры
```

```
&НаСервере
Процедура УстановитьОтбор()
    ПараметрыОтбора = Новый Соответствие();
    ПараметрыОтбора.Вставить("ПоАвтору", ПоАвтору);
    ПараметрыОтбора.Вставить("ПоИсполнителю", ПоИсполнителю);
    УстановитьОтборСписка(Список, ПараметрыОтбора);
КонецПроцедуры
```

Это требование обусловлено тем, что логически процедуры-обработчики событий не предназначены для использования в коде модуля, а вызываются непосредственно платформой. Смещение же этих двух сценариев в одной процедуре неоправданно усложняет ее логику и снижает ее устойчивость (вместо одного предусмотренного сценария вызова - по событию из платформы - код процедуры должен рассчитывать и на другие "прямые" вызовы из кода).

2.5. **Обработчики событий модулей объекта и менеджера объекта** размещаются после раздела с программным интерфейсом, но до служебных процедур и функций модуля.

Методическая рекомендация (полезный совет)

2.5.1. Рекомендуется располагать обработчики, придерживаясь порядка их следования в описании встроенного языка.

2.6. **Служебные процедуры и функции модуля**, которые не являются обработчиками событий, а составляют внутреннюю реализацию модуля, размещаются в модуле следом за обработчиками событий.

В тех случаях когда общий модуль является частью некоторой функциональной подсистемы, включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

Процедуры и функции, связанные между собой по характеру или по логике работы рекомендуется располагать вместе. В модулях форм не рекомендуется явно группировать процедуры и функции модуля на серверные, клиентские и функции без контекста, так как такое «технологическое» упорядочивание затрудняет понимание логики модуля, отвлекая внимание разработчика на детали ее реализации.

2.7. **Раздел инициализации** содержит операторы, инициализирующие переменные модуля или объект (форму).

Например:

```
русск.
#Область Инициализация
АдресПоддержки = "v8@1c.ru";
ВыполнитьИнициализацию();
...

#КонецОбласти
англ.
#Region Initialize
SupportEmail = "v8@1c.ru";
Ctor();
...
#EndRegion
```

для оформления разделов кода в виде областей рекомендуется воспользоваться [приложенной обработкой](#).

Имена процедур и функций

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

#std647

1. Правильный выбор имен процедур и функций очень важен для повышения читаемости кода. В большинстве случаев хорошо выбранное имя процедуры в сочетании с правильно подобранными именами параметров избавляют от необходимости ее как-то дополнительно описывать. В ряде случаев, сложности в выборе имени процедуры и (или) ее параметров

свидетельствуют о неправильной архитектуре программного кода. И наоборот, если "самодокументирующееся" имя придумать легко, значит процедура спроектирована правильно.

См. также: [Описание процедур и функций](#)

2. Имена процедур, функций и [формальных параметров](#) следует образовывать от терминов предметной области таким образом, чтобы из имени было понятно назначение. Следует стремиться к тому, чтобы имена были "говорящими" (документировали сами себя).

Например, неправильно:

Функция ВыполнитьПроверку(Параметр1, Рекв, Т3)

Функция ПолучитьМассивыРеквизитов(ХозяйственнаяОперация, МассивВсехРеквизитов, МассивРеквизитовОперации)

Правильно:

Функция РеквизитОбъектаЗаданногоТипа(Объект, ИмяРеквизита, ТипЗначения)

Функция ЗаполнитьИменаРеквизитовПоХозяйственнойОперации(ХозяйственнаяОперация, ИменаВсеРеквизиты, ИменаРеквизитыОперации)

3. Имена следует образовывать путем удаления пробелов между словами. При этом, каждое слово в имени пишется с прописной буквы. Предлоги и местоимения из одной буквы также пишутся прописными буквами.

Методическая рекомендация (полезный совет)

4. Не рекомендуется в названиях процедур и функций описывать типы принимаемых параметров и (или) возвращаемых значений.

Например, неправильно:

Функция ПолучитьМассивРолейСПравомДобавления()

Функция ПолучитьСтруктуроДополнительныхНастроек()

Правильно:

Функция ИменаРолейСПравомДобавления()

Функция ДополнительныеНастройки()

Эта рекомендация справедлива в большинстве случаев за редким исключением, когда без описания типа возвращаемого значения не ясно назначение самой процедуры или функции.

5. Имена процедур в общем случае, следует образовывать от неопределенной формы глагола, от сути выполняемого действия, например:

Неправильно:

Процедура ЗагрузкаКонтрагента()

Правильно:

Процедура ЗагрузитьКонтрагента()

6.1. Имена функций в общем случае следует образовывать от описания возвращаемого значения.

Неправильно:

Функция ПолучитьПолноеИмяПользователя()

Функция СоздатьПараметрыЗаполненияЦенПоставщика()

Функция ОпределитьДатуНачалаСеанса()

Правильно:

Функция ПолноеИмяПользователя()

Функция НовыеПараметрыЗаполненияЦенПоставщика()

Функция ДатаНачалаСеанса()

6.2. Если функция предназначена для создания какого-либо объекта, то рекомендуется в ее имени использовать слово "Новый". Например,

Неправильно:

Функция ДобавитьПолеФормы()

Функция СоздатьЭлементСправочникаФайлы()

Функция ПолучитьТаблицуКоманд()

Правильно:

Функция НовоеПолеФормы()

Функция НовыйЭлементСправочникаФайлы()

Функция НоваяТаблицаКоманд()

6.3. Если функция выполняет проверку какого-то условия, то ее имя рекомендуется начинать со слова "Это" или использовать причастия.

Неправильно:

Функция ПроверитьПроведенностьДокумента()

Функция ПроверитьИзменениеРеквизитовДокумента()

Функция ВнешняяЗадача()

Правильно:

Функция ДокументПроведен()

Функция РеквизитДокументыИзменены()

Функция ЭтоВнешняяЗадача()

6.4. В имени функции рекомендуется использовать глаголы в неопределенной форме в тех случаях, когда для понимания назначения функции важно, каким образом было получено возвращаемое значение. Например:

Функция ВыбратьДанныеПоПравилу(Правило, ПользовательскиеНастройки)

Функция ПреобразоватьДанныеПоПравилу(НаборыДанных, ПараметрыПреобразования)

6.5. Если выполнение функции предполагает, прежде всего, какое-либо действие, и при этом возврат значения не является ее основной задачей (например, это признак успешно выполненного действия), то имена таких функций следует образовывать от неопределенной формы глагола, как и для процедур.

Например:

Функция РазрешитьРедактированиеРеквизитовОбъекта(Форма)

Описание процедур и функций

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

#std453

1. Описание [процедур и функций](#) рекомендуется выполнять в виде комментария к ним. Необходимость комментирования отдельных участков кода процедур и функций должна определяться разработчиком исходя из сложности и нестандартности конкретного участка кода.

При разработке на платформе 1С:Предприятие 8.3 текст комментария также выводится в контекстной подсказке процедур, функций и их параметров. Подробнее см. раздел «[Контекстная подсказка при вводе текстов модулей](#)» главы 27 «Инструменты разработки» в документации к платформе.

При разработке в 1С:Enterprise Development Tools (EDT) текст комментария также используется для уточнения типизации параметров и возвращаемого значения процедур и функций, и тем самым помогает выявлять ошибки кодирования на этапе разработки.

2. Обязательного комментирования требуют процедуры и функции входящие в **программный интерфейс модулей** - такие процедуры и функции предназначены для использования в других функциональных подсистемах (или в других приложениях), за которые могут отвечать другие разработчики, поэтому они должны быть хорошо документированы.

См. также: [Использование экспортных процедур и функций в модулях форм](#)

3. Прочие процедуры и функции (в том числе обработчики событий модулей форм, объектов, наборов записей, менеджеров значений и т.п.) рекомендуется комментировать, если требуется пояснить назначение процедуры (функции) или особенности её работы. Также рекомендуется описывать причины невыполнения некоторых действий, если они кажутся неочевидными для данной процедуры или функции.

Но если процедура (функция) не сложна для понимания и ее назначение и порядок работы следуют из ее названия и имен формальных параметров, комментарий допускается не писать.

4. Следует избегать комментариев, не дающих дополнительных пояснений о работе не-экспортной процедуры (функции).
Например, неправильно:

```
// Процедура - обработчик события "ПриОткрытии" формы
// &НаКлиенте
Процедура ПриОткрытии()

// Процедура-обработчик команды "Рассчитать"
// &НаКлиенте
Процедура Рассчитать()

// Процедура-обработчик события "ПриИзменении" элемента формы "РедактированиеТолькоВДиалоге"
// &НаКлиенте
Процедура РедактированиеТолькоВДиалогеПриИзменении(Элемент)
```

В этих примерах комментарии избыточны, так как из названий процедур очевидно, что это обработчики событий. А с их описанием и назначением параметров можно ознакомиться в синтакс- помощнике.

```
// Функция возвращает статью движения денежных средств по данным документа
Функция СтатьядвиженияденежныхСредств(ДанныееДокумента)
```

Этот комментарий не дает никакой дополнительной информации о функции.

5. Комментарий размещается перед объявлением процедуры (функции) и имеет следующий вид.

5.1. Секция "Описание" (англ. "Description") содержит описание назначения процедуры (функции), достаточное для понимания сценария ее использования без просмотра ее исходного кода. Так же может содержать краткое описание принципов работы и перекрестные ссылки на связанные процедуры и функции.

Может быть единственной секцией для процедур без параметров. Описание не должно совпадать с именем процедуры (функции). Для процедур и функций секция должна начинаться с глагола. Для функций это, как правило: «Возвращает...». В тех случаях, когда возвращаемый результат является не основным в работе функции, – то с основного действия, например: «Проверяет...», «Сравнивает...», «Вычисляет...» и т.п. Не рекомендуется начинать описание с избыточных слов «Процедура...», «Функция...», а также с именем самой процедуры (функции), от удаления которых смысл не меняется.

Например, неправильно:

```
// Конструктор объекта WSПрокси.
// ...
Функция WSПрокси(ПараметрыПрокси) Экспорт

// Функция СтрокаТаблицыЗначенийВСтруктуру создает структуру со свойствами, соответствующими...
Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт
```

Правильно:

```
// Создает прокси на основе определения веб-сервиса и связывает
// его с точкой подключения веб-сервиса.
// В дополнении к платформенному конструктору Новый WSПрокси:
//   - включает в себя вызов конструктора WSОпределения;
//   - на время сеанса кширует файл WSDL для оптимизации частых обращений к веб-сервису;
//   - не требует явного указания ИнтернетПрокси (он подставляется автоматически, если настроен);
//   - выполняет быструю проверку доступности веб-сервиса с помощью операции Ping.
// ...
Функция WSПрокси(ПараметрыПрокси) Экспорт
```

```
// Создает структуру со свойствами, соответствующими...
Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт
```

5.2. Секция "Параметры" (англ. "Parameters") описывает [параметры процедуры \(функции\)](#). Если их нет, секция пропускается. Предваряется строкой "Параметры:", затем с новой строки размещаются описания всех параметров.

5.2.1. Описание параметра начинается с новой строки, далее имя параметра, затем дефис и список типов (*), далее дефис и текстовое описание параметра.

Имя параметра необходимо стремиться выбирать таким образом, чтобы его назначение было понятно в контексте функции без дополнительных пояснений

Описание типа является обязательным. Тип может быть описан явно, при этом может быть указан или один тип или список типов. Под «списком типов» подразумеваются имена типов, разделенные запятыми. Имя типа может быть простым (в одно слово) или составным - в два слова, разделенных точкой.

Например: Стока, Структура, Произвольный, СправочникСсылка.Сотрудники.

В качестве типов значений следует использовать только существующие в платформе типы, а также специальные типы, предусмотренные в EDT: **ОпределяемыйТип.<Имя>**, **СправочникСсылка**, **ОбъектМетаданныхОтчет**, **РасширениеДекорацииФормыДляНадписи** и т.п.

Например, неправильно:

```
// КоллекцияСтрок - КоллекцияЗначений – коллекция для сравнения;
// ФормируемыйОтчет - ОбъектМетаданных: Отчет
// ПрисоединенныйФайлОбъект - элемент справочника файлов.
```

Правильно:

```
// КоллекцияСтрок – ТаблицаЗначений, Массив, СписокЗначений – коллекция для сравнения.
// ФормируемыйОтчет – ОбъектМетаданныхОтчет
// ПрисоединенныйФайлОбъект – ОпределяемыйТип.ПрисоединенныйФайлОбъект – элемент справочника файлов.
```

Текстовое описание параметра рекомендуется заполнять в том случае, когда только имени параметра в контексте функции не достаточно для понимания его назначения, либо требуется дать дополнительную информацию о типе, поясняющие назначение параметра, а также может приводиться наглядный пример с ожидаемым значением параметра.

Например, неправильно:

```
// Проверяет, что переданные адреса включены в задачу. Если проверка не проходит – генерируется исключение.
// Параметры:
```

```
// Адреса - Стока - строка, содержащая электронные адреса  
// ЗадачаИсполнителя - ЗадачаСсылка.ЗадачаИсполнителя – проверяемая задача  
//  
Процедура ПроверитьАдресаЗадачи(Адреса, ЗадачаИсполнителя)
```

Правильно:

```
// Проверяет, что переданные адреса включены в задачу. Если проверка не проходит – генерируется исключение.  
//  
// Параметры:  
// Адреса - Стока - содержит электронные адреса, разделенные запятой. Например, support@mycorp.ru,v8@localdomain.  
// ЗадачаИсполнителя - ЗадачаСсылка.ЗадачаИсполнителя  
//  
Процедура ПроверитьАдресаЗадачи(Адреса, ЗадачаИсполнителя)
```

В данном примере текстовое описание для параметра «Адреса» нужно чтобы

- указать правило передачи нескольких адресов (через запятую);
- привести пример.

Текстовое описание для параметра **ЗадачаИсполнителя** не нужно.

5.2.2. Для параметров типа **Структура** и **ТаблицаЗначений** также задается описание их свойств и колонок, которые начинаются с новой строки и предваряются символом *.

Например:

```
// Параметры:  
// СтатусыСерий - ТаблицаЗначений:  
//   * Серия - СправочникСсылка.СерииНоменклатуры - если серия указана и она может  
//     использоваться с новым значением номенклатуры на указанном складе,  
//     то возвращается переданное значение; если нет - пустая ссылка  
//   * СтатусУказанияСерий - Число - если серии указываются в ТЧ "Товары", то  
//     возвращается рассчитанный статус, если для переданной  
//     номенклатуры/склада серии не используется - возвращается 0  
//     иначе возвращается переданный статус.
```

При этом текстовое описание свойства (колонки) рекомендуется заполнять в том случае, когда только имени свойства в контексте параметра не достаточно для понимания назначения свойства или требуется дать дополнительную информацию о типе.

5.2.3. Для параметров типа **Массив** следует указывать тип элементов с помощью ключевого слова "из" (англ. "of"):

Например, неправильно:

```
// МассивПеренаправленныхЗадач - Массив - массив перенаправленных задач.  
// МассивПеренаправленныхЗадач - Массив - задачи (ЗадачаСсылка.ЗадачаИсполнителя), перенаправленные другому исполнителю.
```

Правильно:

```
// ПеренаправленныеЗадачи - Массив из ЗадачаСсылка.ЗадачаИсполнителя
```

```
// СведенияОбОбновлении - Массив из Структура:  
//   * КодАдресногоОбъекта - Стока  
//   * Наименование - Стока  
//   * Индекс - Стока  
//   * Обновление доступно - Булево  
//
```

В описании массивов, структур и таблиц значений могут быть вложенные описания, при этом перед именами вложенных свойств число звездочек увеличивается: для первого уровня вложенности 2 звездочки, для второго 3 и т.д.

5.2.4. Также для параметра типа **СтроКТаблицыЗначений** (**СтроДереваЗначений**) возможно задать состав свойств, соответствующий колонкам его таблицы-владельца (дерева-владельца):

Например:

```
// СведенияОРегионе – СтроКТаблицыЗначений: см. РегистрыСведений.АдресныеОбъекты.КлассификаторСубъектовРФ
```

где **КлассификаторСубъектовРФ** – экспортная функция модуля менеджера регистра сведений АдресныеОбъекты, которая возвращает таблицу значений.

5.2.5. Также для каждого параметра можно задать одно или несколько дополнительных описаний типов параметра. Каждое дополнительное описание начинается с новой строки, затем обязательный дефис, далее список типов параметра далее дефис и текстовое описание.

Например:

```
// Параметры:  
//   Реквизиты - Стока - имена реквизитов, перечисленные через запятую.  
//     Например, "Код, Наименование, Родитель".  
//   - Структура, ФиксированнаяСтруктура - в качестве ключа передается  
//     псевдоним поля для возвращаемой структуры с результатом,  
//     а в качестве значения (опционально) фактическое имя поля в таблице.  
//     Если значение не определено, то имя поля берется из ключа.  
//   - Массив Из Стока, ФиксированныйМассив Из Стока - имена реквизитов.
```

5.2.6. Описание также могут быть заданы с помощью ссылки на [функцию-конструктор](#) в формате "см. ПутьКФункции" (англ "see MethodPath").

Например:

```
// ПараметрыУказанияСерий - см. НоменклатураКлиентСервер.ПараметрыУказанияСерий  
// Дубли - см. ОбработкаОбъект.ПоискиУдалениеДублей.ГруппыДублей  
// РеквизитыКомпонент - Массив из см. ВнешниеКомпоненты.РеквизитыКомпоненты
```

При разработке кода, обращающегося к реквизитам конкретного объекта метаданных или формы, можно ссылаться на типы реквизитов этого объекта (формы):

```
// Запросы - см. Обработка.КонсольЗапросов.ТабличнаяЧасть.Запросы  
// ТипыДанных - см. Обработка.КонсольЗапросов.Реквизит.ДоступныеТипыДанных  
// Вложения - см. Справочники.ШаблоныСообщений.ФормаЕлемента.Вложения  
// КонтактнаяИнформация - см. Документы.ЗаказПокупателя.ФормаДокумента.Объект.КонтактнаяИнформация
```

Также в редких случаях, когда подходящей функции-конструктора не существует и ее невозможно создать, допустимо указывать ссылку на другую процедуру (при полном совпадении параметров) или на параметр другой процедуры или функции, например:

```

// См. ПодключаемыеКомандыПереопределяемый.ПриОпределенииКомандПодключенныхКОбъекту
// Процедура ПриОпределенииКомандПодключенныхКОбъекту(НастройкиФормы, Источники, ПодключенныеОтчетыИОбработки, Команды) Экспорт

// Параметры:
// НастройкиФормы - см. ПодключаемыеКомандыПереопределяемый.ПриОпределенииКомандПодключенныхКОбъекту.НастройкиФормы
// Источники - см. ПодключаемыеКомандыПереопределяемый.ПриОпределенииКомандПодключенныхКОбъекту.Источники
// ПодключенныеОтчетыИОбработки - см.
ПодключаемыеКомандыПереопределяемый.ПриОпределенииКомандПодключенныхКОбъекту.ПодключенныеОтчетыИОбработки
// Команды - см. ПодключаемыеКомандыПереопределяемый.ПриОпределенииКомандПодключенныхКОбъекту.Команды
// Процедура ПриОпределенииКомандПодключенныхКОбъекту(НастройкиФормы, Источники, ПодключенныеОтчетыИОбработки, Команды) Экспорт

5.3. Секция "Возвращаемое значение" (англ. "Returns") описывает тип и содержание возвращаемого значения функции. Для процедур эта секция отсутствует. Предваряется строкой "Возвращаемое значение:". Затем с новой строки тип возвращаемого значения, дефис и текст описания. При использовании возвращаемого значения составного типа следует каждый тип писать с новой строки и с дефиса. Например:

// Возвращаемое значение:
// Стока

// Возвращаемое значение:
// Булево - Истина, если хотя бы одна из переданных ролей доступна текущему пользователю, либо у него есть административные права.

// Возвращаемое значение:
// - ЛюбаяСсылка - ссылка на предопределенный элемент.
// - Неопределено - если предопределенный элемент есть в метаданных, но не создан в ИБ.

// Возвращаемое значение:
// - СправочникСсылка.Пользователи
// - СправочникСсылка.ВнешниеПользователи

Текстовое описание возвращаемого значения рекомендуется заполнять в том случае, когда только одного описания функции не достаточно, либо требуется дать дополнительную информацию о типе, например, о составе свойств или колонок возвращаемого значения. Также может быть приведен пример с ожидаемым значением возвращаемого значения, либо сквозной пример размещается в секции "Пример" ниже.

Для возвращаемых значений также действуют требования п.5.2.2 и 5.2.3.

5.4. Секция "Пример" (англ. "Example") содержит пример использования процедуры, или функции. Предваряется строкой "Пример:". Далее с новой строки пример использования. Имя процедуры (функции) следует писать вместе с именем общего модуля, в котором она расположена. Из примера должно быть понятно, что передается на входе и что возвращается на выходе. Например, неправильно:

// Пример:
// ПодставитьПараметрыВСтроку(ШаблонСтроки, СтрокаЗамены);

Правильно:

// Пример:
// СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("гу='%1 пошел в %2"), "Вася", "Зоопарк") = "Вася пошел в Зоопарк".

предопределенных модулях в секции "Пример" следует размещать пример реализации переопределяемой процедуры, а не пример ее вызова. Например, для процедуры ПриОпределенииОбщихПараметровБазовойФункциональности(ОбщиеПараметры):

// Пример:
// ОбщиеПараметры.МинимальноНеобходимаяВерсияПлатформы = "8.3.4.365";
// ОбщиеПараметры.РекомендуемыйОбъемОперативнойПамяти = 2;

5.5. В редких случаях, когда сразу несколько параметров имеют дополнительные типы, рекомендуется добавить секцию "Варианты вызова" (англ. "Call options"), в которой дать описания наиболее частых или всех возможных вариантов вызова функции с различными комбинациями типов параметров. Секция начинается фразой "Варианты вызова:" с новой строки, затем идут описания вариантов, каждое начинается с новой строки. Каждый вариант вызова представляется в виде имени функции со списком типов, перечисленных через запятую в круглых скобках, затем следует дефис и текстовое описание варианта.

Например:

// ...
// Параметры:
// Параметр1 - Тип11, Тип12 - ...
// Параметр2 - Тип21, Тип22, Тип23 - ...
//
// Варианты вызова:
// УниверсальнаяПроцедура(Тип11, Тип21) - описание ...
// УниверсальнаяПроцедура(Тип12, Тип22) - описание ...
// УниверсальнаяПроцедура(Тип11, Тип23) - описание ...
//
Процедура УниверсальнаяПроцедура(Параметр1, Параметр2) Экспорт

5.6. В любом месте документирующего комментария можно добавить переход к другим объектам конфигурации, процедурам и функциям (в частности, для перехода к функциям-конструкторам структур). При использовании 1С:Enterprise Development Tools среда оформит такие переходы в виде гиперссылки.

Например:

// Описание универсальной процедуры.
// См. Управление доступом.ЗаполнитьНаборыЗначенийДоступа
//
// Параметры:
// Параметр1 - Произвольный - описание параметра см. Справочник.Контрагенты.
//
Процедура УниверсальнаяПроцедура(Параметр1)

5.7. В случаях когда возникает необходимость отметить процедуру (функцию) как устаревшую, в первой строке ее описания размещается слово "Устарела" (англ. "Deprecated")..
Например:

// Устарела. Следует использовать новую см. ОбщегоНазначения.ЕстьРоль
// ...
Функция Ролидоступны(ИменаРолей) Экспорт

6. Если требуется прокомментировать процедуру или функцию с директивой компиляции, то вначале следует размещать комментарий, а затем - директиву компиляции. Например:

// Процедура - обработчик события формы ПриСозданииНаСервере.
// Обрабатывает параметры формы и заполняет реквизиты формы значениями.
// А также выполняет следующие действия:
// ...
//
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

```

Такой стиль размещения комментария позволяет в первую очередь обращать внимание на определение функции и директиву компиляции, а потом - на комментарий, который может занимать достаточно большое количество строк.

7. Код процедур и функций должен отделяться друг от друга в тексте модуля пустыми строками.

Примеры описания процедур и функций

Пример описания функции с одним параметром:

```
// Определяет доступность ролей ИменаРолей текущему пользователю,
// а также доступность административных прав.
//
// Параметры:
//   ИменаРолей - Стока - имена ролей, доступность которых проверяется, разделенные запятыми.
//
// Возвращаемое значение:
//   Булево - Истина, если хотя бы одна из переданных ролей доступна текущему пользователю,
//   либо у него есть административные права.
//
// Пример:
//   Если Ролидоступны("ИспользованиеРассылокОтчетов,ОтправкаПоПочте") Тогда ...
//
Функция Ролидоступны(ИменаРолей) Экспорт
```

Пример описания процедуры без параметров:

```
// В обработчике события ПередЗаписью документа выполняется;
//   - очистка табличной части услуги, в случае если указан договор с комиссионером;
//   - проверка заполнения реквизита ЕдиницаИзмеренияМест табл. части Товары;
//   - синхронизация с "подчиненным" счетом-фактурой;
//   - заполнение склада и заказа покупателя в табличных частях Товары и ВозвратнаяТара;
//   - удаление неиспользуемых строк табличной части "Серийные номера";
//   - заполнение переменной модуля объекта УдалятьДвижение.
```

Процедура ПередЗаписью()

КонецПроцедуры

Для автоматического упорядочивания комментариев к процедурам или функциям с директивами компиляции можно воспользоваться приложенной обработкой [ФорматированиеДирективКомпиляции.ерп](#).

Для этого необходимо:

1. Выгрузить модули конфигурации (команда меню Конфигурация -> Выгрузить файлы конфигурации...)
2. Открыть обработку в режиме **1С:Предприятие** и указать каталог, в который были выгружены модули - далее нажать кнопку "Форматировать"
3. Загрузить модули в конфигурацию (команда меню Конфигурация -> Загрузить файлы конфигурации...)

Параметры процедур и функций

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

#std640

1. При объявлении формальных параметров процедур и функций (далее по тексту: функций) необходимо придерживаться [общих правил образования имен переменных](#). В частности, имена параметров следует образовывать от терминов предметной области таким образом, чтобы из имени параметра было понятно его назначение.

2. Не следует использовать вместо параметров функций другие средства конфигурирования ([переменные модулей](#), реквизиты формы и т.п.)

3. Параметры в функции должны идти в логической последовательности. Рекомендуется располагать параметры по принципу от общего к частному.
Например, неправильно:

Процедура ПересчитатьСуммуДокумента(ИмяПоляСумма, ДокументОбъект, СуммаВключаетНДС = Истина)

Процедура ПоменятьЦветПоляФормы(Цвет, ИмяПоля, Форма)

правильно сначала расположить основные параметры **ДокументОбъект** и **Форма**:

Процедура ПересчитатьСуммуДокумента(ДокументОбъект, ИмяПоляСумма, СуммаВключаетНДС = Истина)

Процедура ПоменятьЦветПоляФормы(Форма, ИмяПоля, Цвет)

4. Необязательные параметры (параметры со значениями по умолчанию) должны располагаться после обязательных параметров (без значений по умолчанию).
Например:

Функция КурсВалютыНадату(Валюта, Дата = Неопределено) Экспорт

5. Не рекомендуется объявлять в функциях много параметров (нужно ориентироваться на количество не более семи параметров), при этом не должно быть много параметров со значениями по умолчанию (нужно ориентироваться на количество не более трех таких параметров). В противном случае, читаемость вызывающего кода сильно снижается. Например, можно легко ошибиться в количестве запятых при передаче необязательных параметров.

При необходимости передавать в функцию большое число параметров рекомендуется:

- группировать однотипные параметры в один или несколько составных параметров типа **Структура**. Например, в структуры могут быть объединены параметры, описывающие состав и значения полей некоторого объекта (**ДанныеЗаполнения**, **ПараметрыПроведения**, **ДанныеФайла** и т.п.);
- либо полностью пересмотреть логику работы функции, например, разделив ее на несколько разных, более простых функций.

Например, неправильно:

```
// Добавляет новое поле на форму, инициализирует его значениями по умолчанию.
```

```
Функция ДобавитьПолеФормы(ИмяПоля, Заголовок = Неопределено, ОбработчикНачалоВыбора = "", ШиринаПоля,
ЦветФона = Неопределено, ЦветФонаЗаголовка = Неопределено, Родитель = Неопределено, КартинкаШапки = Неопределено, ПутьКДанным =
ТолькоПросмотрПоля = Ложь, СвязиПараметровВыбора = Неопределено)
```

...

КонецФункции

// вызывающий код

```
НовоеПоле = ДобавитьПолеФормы("СтараяЦена", НСтр("гру='Цена'"),,, 12, ЦветФона, ЦветЗаголовка, НоваяГруппа,,, Истина);
НовоеПоле.ЦветТекста = WebЦвета.Серый;
```

Правильно пересмотреть логику работы функций, оставив в ней только один ключевой параметр **ИмяПоля**:

```
// Добавляет новое поле на форму, инициализирует его значениями по умолчанию.
```

```
Функция НовоеПолеФормы(ИмяПоля)
```

...
КонецФункции

```
// вызывающий код  
НовоеПоле = НовоеПолеФормы("СтараяЦена");  
НовоеПоле.Заголовок = НСтр("ги='Цена'");  
НовоеПоле.ЦветФона = ЦветФона;  
НовоеПоле.ЦветТекста = WebЦвета.Серый;  
НовоеПоле.... = ...  
...
```

Другой пример. Неправильно:

```
// Создает элемент справочника "Номенклатура"  
Процедура СоздатьЭлементНоменклатуры(Наименование, ТоварУслуга, ЕдиницаИзмерения, ВесНетто, ПроверятьУникальность = Истина)  
...  
КонецПроцедуры
```

Правильно сгруппировать параметры, описывающие значения реквизитов номенклатуры, в структуру **ЗначенияРеквизитов**:

```
// Создает элемент справочника "Номенклатура"  
Процедура СоздатьЭлементНоменклатуры(ЗначенияРеквизитов, ПроверятьУникальность = Истина)  
...  
КонецПроцедуры
```

6. При вызове функций необходимо избегать громоздких конструкций, которые приводят к снижению читаемости кода, увеличивают вероятность ошибок и затрудняют отладку.
В частности:

6.1. Не рекомендуется при передаче параметров в одну функцию применять вложенные вызовы других функций.
Неправильно:

```
СтруктураВложений.Вставить(ПрисоединенныйФайл.Наименование,  
Новый Картинка(ПолучитьИзВременногоХранилища(ПрисоединенныеФайлы.ПолучитьДанныеФайла(ПрисоединенныйФайл.Ссылка).СсылкаНадвоичныеДанные));
```

Правильно разбивать такие вызовы на отдельные операторы с помощью вспомогательных локальных переменных:

```
АдресФайлаИзображения = ПрисоединенныеФайлы.ПолучитьДанныеФайла(ПрисоединенныйФайл.Ссылка).СсылкаНадвоичныеДанныеФайла;  
ДанныеИзображения = Новый Картинка(ПолучитьИзВременногоХранилища(АдресФайлаИзображения));  
СтруктураВложений.Вставить(ПрисоединенныйФайл.Наименование, ДанныеИзображения);
```

В то же время, если код с вложенными вызовами получается компактным (не требует переноса выражений) и легко читаемым, то вложенные вызовы допустимы.
Например:

```
Предупреждение(НСтр("ги='Для выполнения операции необходимо установить расширение работы с файлами.'"));
```

ПеречитатьСуммуПоКурсу(Сумма, КурсВалютыНадату(Валюта));

6.2. Также не рекомендуется при вызове функций использовать вложенный конструктор структуры: **Новый Структура(...)**. Вложенное объявление структуры допустимо только в тех случаях, когда количество ее свойств небольшое (нужно ориентироваться на количество свойств не более трех).

Неправильно:

```
ЗаполнитьЦены(Объект.Товары, , Новый Структура("Дата, Валюта, Соглашение, ПоляЗаполнения",  
Объект.Дата, Объект.Валюта, Объект.Соглашение, "Цена, СтавкаНДС, ВидЦены, СрокПоставки"),  
Новый Структура("ПересчитатьСумму, ПересчитатьСуммуНДС, ПересчитатьСуммуРучнойСкидки, ОчиститьАвтоматическуюСкидку",  
"КоличествоУпаковок", СтруктураПересчетаСуммы, СтруктуроПересчетаСуммы, "КоличествоУпаковок", Неопределенено, Неопределенено));
```

Правильно:

```
ПараметрыЗаполнения = Новый Структура;  
ПараметрыЗаполнения.Вставить("Дата", Объект.Дата);  
ПараметрыЗаполнения.Вставить("Валюта", Объект.Валюта);  
ПараметрыЗаполнения.Вставить("Соглашение", Объект.Соглашение);  
ПараметрыЗаполнения.Вставить("ПоляЗаполнения", "Цена, СтавкаНДС, ВидЦены, СрокПоставки");
```

```
ДействияСИзмененнымиСтрокаами = Новый Структура;  
ДействияСИзмененнымиСтрокаами.Вставить("ПересчитатьСумму", "КоличествоУпаковок");  
ДействияСИзмененнымиСтрокаами.Вставить("ПересчитатьСуммуНДС", ПараметрыПересчетыСуммы);  
ДействияСИзмененнымиСтрокаами.Вставить("ПересчитатьСуммуРучнойСкидки", ПараметрыПересчетыСуммы);  
ДействияСИзмененнымиСтрокаами.Вставить("ПересчитатьСуммуРучнойСкидки", "КоличествоУпаковок");  
ДействияСИзмененнымиСтрокаами.Вставить("ОчиститьАвтоматическуюСкидку");  
ДействияСИзмененнымиСтрокаами.Вставить("ОчиститьСуммуВзаиморасчетов");
```

ЗаполнитьЦены(Объект.Товары, ПараметрыЗаполнения, ДействияСИзмененнымиСтрокаами);

7. При вызове функций не следует пропускать обязательные параметры. В противном случае, в параметр будет передано значение **Неопределено**, на которое функция может быть не рассчитана. Если же значение **Неопределено** является допустимым, то нужно или его передавать в функцию явно, или сделать этот параметр необязательным со значением по умолчанию **Неопределено**.

Например, для вызова процедуры

```
Процедура ПоменятьЦветПоляФормы(Форма, ИмяПоля, Цвет)
```

неправильно:

```
ПоменятьЦветПоляФормы(, "РезультатПроверки", ЦветаСтиля.ПоясняющийОшибкаТекст); // пропущен первый параметр Форма  
ПоменятьЦветПоляФормы(, , ); // пропущены все обязательные параметры
```

правильно:

```
ПоменятьЦветПоляФормы(ЭтотОбъект, "РезультатПроверки", Цвет); // указаны все обязательные параметры
```

См. также

- [Передача параметров по ссылке и по значению при вызове процедур и функций](#) (статья на ИТС)
- [Особенности использования структур в качестве параметров процедур и функций](#)
- [Использование объектов типа Структура](#)

Особенности использования структур в качестве параметров процедур и функций

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

#std641

Основная статья: [Параметры процедур и функций](#)

Для процедур и функций (далее по тексту: функций) с параметрами типа **Структура** рекомендуется придерживаться следующего подхода к разработке.

1. Помимо функции, которая собственно реализует прикладную функциональность (далее по тексту: вызываемая функция), необходимо определить функция-конструктор для создания новой структуры (далее по тексту функция-конструктор параметров). При этом сама функция-конструктор не принимает параметров, а только возвращает структуру-заготовку со свойствами, которую вызывающий код должен проинициализировать конкретными значениями и передать в вызываемую функцию.

Пример вызывающего кода:

Процедура ПриИзменениеНоменклатурыСервер(ИдентификаторТекущейСтроки)

```
// Получаем новую структуру параметров.  
ПараметрыЗаполненияЦен = ЦенообразованиеКлиентСервер.ПараметрыЗаполненияЦеныВСтроКетЧ();  
// Заполняем параметры.  
ПараметрыЗаполненияЦен.Дата = Объект.Дата;  
ПараметрыЗаполненияЦен.Валюта = Объект.Валюта;  
  
ТекущаяСтрока = Объект.Товары.НайтиПоИдентификатору(ИдентификаторТекущейСтроки);  
  
// Передаем структуру параметров в прикладную функцию.  
ЦенообразованиеСервер.ЗаполнитьЦеныВСтроКетЧ(ТекущаяСтрока, ПараметрыЗаполненияЦен);
```

КонецПроцедуры

Пример функции-конструктора параметров в модуле **ЦенообразованиеКлиентСервер**:

Функция ПараметрыЗаполненияЦеныВСтроКетЧ() Экспорт

```
ПараметрыЗаполненияЦен = Новый Структура;  
ПараметрыЗаполненияЦен.Вставить("Дата");  
ПараметрыЗаполненияЦен.Вставить("Валюта");  
ПараметрыЗаполненияЦен.Вставить("ПересчитыватьСумму", Истина);  
ПараметрыЗаполненияЦен.Вставить("ОбязательныеПараметры", "Дата, Валюта"); // обязательные параметры, которые нужно заполнять  
Возврат ПараметрыЗаполненияЦен;
```

КонецФункции

Имена свойств структуры соответствуют параметрам вызываемой функции. При этом параметры со значениями по умолчанию должны быть явно проинициализированы в этой структуре.

2. В вызывающем коде не следует инициализировать структуру параметров или добавлять в нее какие-либо другие свойства. Во избежание неоднозначности и скрытых ошибок все допустимые параметры вызываемой функции должны быть определены явно в функции-конструкторе параметров.

См. также

- [Использование объектов типа Структура](#)

Правила образования имен переменных

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

#std454

1. Имена переменных следует образовывать от терминов предметной области таким образом, чтобы из имени переменной было понятно ее назначение.

2. Имена следует образовывать путем удаления пробелов между словами. При этом, каждое слово в имени пишется с прописной буквы. Предлоги и местоимения из одной буквы также пишутся прописными буквами.

Пример:

```
Перем ДиалогРаботыСКаталогом; // Диалог работы с каталогом  
Перем КоличествоПачекВКоробке; // Количество пачек в коробке
```

Примеры некорректных имен переменных:

масРеквизитов, соотвВидИмя, новСтр

3. Имена переменных запрещается начинать с подчеркивания.

4. Имена переменных не должны состоять из одного символа. Использование односимвольных имен переменных допускается только для счетчиков циклов.

5. Переменные, отражающие состояние некоторого флага, следует называть так, как пишется истинное значение этого флага.

Например:

```
Перем ЕстьОшибки; // Признак наличия ошибок в процедуре.  
Перем ЭтоТоварТара; // Признак, что товар относится к возвратной таре.
```

См. также

- [Использование глобальных переменных в программных модулях](#)

Работа с параметром «Отказ» в обработчиках событий

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

#std686

1. В обработчиках событий модулей объектов, наборов записей, форм и т.п., содержащих параметр **Отказ** (ПриЗаписи, ОбработкаПроверкиЗаполнения, ТоварыПередНачаломДобавления и т.п.), не следует присваивать этому параметру значение **Ложь**.

Это требование обусловлено тем, что, как правило, в коде обработчиков событий параметр **Отказ** может устанавливаться сразу в нескольких последовательных проверках (или в нескольких подписках на одно и то же событие). В таком случае к моменту выполнения очередной проверки параметр **Отказ** уже может заранее содержать значение **Истина**, и можно ошибочно сбросить его обратно в **Ложь**.

Кроме того, при доработках конфигурации на внедрении число этих проверок может увеличиться.

Неправильно:

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)  
...  
Отказ = ЕстьОшибкиЗаполнения();  
...  
КонецПроцедуры
```

Правильно:

```
Процедура ОбработкаПроверкиЗаполнения(Отказ, ПроверяемыеРеквизиты)  
...
```

```
Если ЕстьОшибкаЗаполнения() Тогда
    Отказ = Истина;
КонецЕсли;
...
КонецПроцедуры
```

или

Отказ = Отказ Или ЕстьОшибкаЗаполнения();

2. Эти же требования справедливы для других аналогичных параметров обработчиков событий: **СтандартнаяОбработка**, **Выполнение** и др.
Например:

```
Процедура ОбработкаПолученияДанныхВыбора(данныеВыбора, Параметры, СтандартнаяОбработка)
```

```
Если Параметры.Свойства(...) Тогда
    СтандартнаяОбработка = Ложь;
```

...

```
КонецЕсли;
```

```
КонецПроцедуры
```

3. Неприемлемо в событиях объекта **ПриЗаписи**, **ПередЗаписью**, **ПередУдалением**, **ОбработкаПроведения**, **ОбработкаПроверкиЗаполнения** и т.п устанавливать параметр **Отказ** в значение **Истина** без информирования пользователя о причинах:

Неправильно:

Процедура ОбработкаПроведения(Отказ, РежимПроведения)

```
Если Не ТоваровНаСкладедостаточно() Тогда
    Отказ = Истина;
    Возврат;
КонецЕсли;
```

...

```
КонецПроцедуры
```

В таких случаях сообщение об ошибке не информативно и скрывает истинные причины проблемы, поэтому расследование ошибки требует значительных временных затрат:

Не удалось записать "Заказ покупателя"!

Следует согласно пп. 1.1 и 1.3 стандарта [Информирование пользователя](#) корректно уведомить пользователя о причинах отказа с помощью сообщения или вызова исключения.

Правильно:

Процедура ОбработкаПроведения(Отказ, РежимПроведения)

```
Если Не ТоваровНаСкладедостаточно() Тогда
```

```
    ТекстСообщения = СтрШаблон(НСтр("ru='Не хватает %1 %2 товара %3, на складе %4'"), Количество, ЕдиницаИзмерения, Товар, Склад);
    ОбщегоНазначения.СообщитьПользователю(ТекстСообщения,, "Объект.Товары",, Отказ);
```

```
КонецЕсли;
```

...

```
КонецПроцедуры
```

См. также

- [Информирование пользователя](#)
- [Перехват исключений в коде](#)
- [Сообщения пользователю](#)

Общие требования к построению конструкций встроенного языка

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

#std441

1. В конструкциях встроенного языка ключевые слова пишутся канонически (как в документации или Синтакс-помощнике).
Правильно:

КонецЕсли

Неправильно:

конецЕсли, КОНЕЦЕСЛИ, конецесли, Конечесли.

2. При следовании друг за другом нескольких операторов присваивания, допускается выравнивать их следующим образом:

```
ДиалогВыбора.ПолноеИмяФайла = ИмяФайла;
ДиалогВыбора.Каталог      = ИмяПути;
ДиалогВыбора.Заголовок     = НСтр("ru = 'Выберите файл со списком запросов'");
ДиалогВыбора.Фильтр       = НСтр("ru = 'Файлы запросов (*.sel)|*.sel|Все файлы (*.*)|*.*'");
ДиалогВыбора.Расширение   = "sel";
```

При этом не следует выравнивать операторы одинаково по всему модулю - рекомендуется делать выравнивание только для операторов, расположенных рядом.

3. Составные логические выражения в **Если..КонецЕсли** переносятся согласно [правилам переноса выражений](#).

4. Логические выражения и логические значения (например, результат функции, возвращающей логическое значение, переменные типа **Булево** и пр.) не следует проверять путем сравнения с литералами **Истина** и **Ложь**.

Правильно:

Если ЭтоНовый() Тогда

Неправильно:

Если ЭтоНовый() = Истина Тогда

5. В тех случаях, когда требуется сравнивать результаты каких-либо выражений, следует предварительно присваивать результаты этих выражений промежуточным переменным, и сравнивать уже сами эти переменные.

Правильно:

```
Ответ = Вопрос(НСтр("ru = 'данные еще не записаны. Записать?'"), РежимДиалогаВопрос.ДаНет,, КодВозвратаДиалога.Да);
Если Ответ = КодВозвратаДиалога.Да Тогда
    Записать();
Иначе
    Возврат;
КонецЕсли;
```

Неправильно:

```
Если Вопрос(НСтр("ru = 'данные еще не записаны. Записать?'"), РежимДиалогаВопрос.ДаНет,, КодВозвратаДиалога.Да) =
КодВозвратаДиалога.Да Тогда
    Записать();
Иначе
    Возврат;
КонецЕсли;
```

6. Необходимо использовать системные наборы значений везде, где возможно их применить, например, вместо **Символ(10)** следует использовать **Символы.ПС.**

См. также

- [Перенос выражений](#)
- [Тексты модулей](#)

Перенос выражений

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

#std444

1. При длине строки более 120 символов следует использовать переносы. Строки длиннее 120 символов делать не рекомендуется, за исключением тех случаев, когда перенос невозможен.

2. Длинные арифметические выражения переносятся следующим образом:

- в одной строке может находиться более одного операнда;
- при переносе знаки операции пишутся в начале строки (а не в конце предыдущей строки);
- операнды на новой строке предваряются стандартным отступом, либо выравниваются по началу первого операнда без учета знаков операций.

Пример:

```
СуммаДокумента = СуммаБезСкидки
    + СуммаРучнойСкидки
    + СуммаАвтоматическойСкидки;
```

или

```
СуммаДокумента = СуммаБезСкидки
    + СуммаРучнойСкидки
    + СуммаАвтоматическойСкидки;
```

3.1 Длинные строковые константы рекомендуется переносить с помощью специального символа перевода на новую строку, например:

```
ТекстЗапроса =
"ВЫБРАТЬ РАЗРЕШЕННЫЕ
| ЗаметкиПоПредмету.КоличествоЗаметок КАК КоличествоЗаметок
|ИЗ
| РегистрСведений.ЗаметкиПоПредмету КАК ЗаметкиПоПредмету
|ГДЕ
| ЗаметкиПоПредмету.Предмет = &Предмет";
```

или

```
ТекстПредупреждения = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
    НСтр("ru = 'Обновление адресного классификатора не требуется.
        |В программе уже загружены актуальные адресные сведения от %1.'"),
    Формат(датаПоследногоОбновленияЯЛДР, "ДДФ=Д"));
ПоказатьПредупреждение(, ТекстПредупреждения);
```

При этом не следует переносить строки, содержащие текст сообщения пользователю (объект **СообщениеПользователю**).

3.2. В общем случае при конкатенации строк знак "+" рекомендуется писать в начале строки, так же как и при переносе арифметических выражений (см. п.2), например:

```
ПоляОтбора = "Номенклатура, Характеристика, Склад"
    + ДополнительныеПоляОтбора;
```

3.3. При конкатенации длинных строк знак "+" можно писать в конце строки, чтобы не ломать общее форматирование текста. Например,

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

4. При необходимости параметры процедур, функций и методов следует переносить следующим образом:

- параметры выравниваются по началу первого параметра, либо предваряются стандартным отступом;
- закрывающая скобка и разделитель операторов ";" пишется в той же строке, что и последний параметр;
- также допустим и способ форматирования, который предлагает функция автоформатирования в конфигураторе (см. п. 5).

Пример:

```
ИменаДокументов = Новый СписокЗначений;
ИменаДокументов.Добавить(Метаданные.Документы.СтрокаВыпискиРасход.Имя,
    Метаданные.Документы.СтрокаВыпискиРасход.Синоним);
ИменаДокументов.Добавить(Метаданные.Документы.РасходныйКассовыйОрдер.Имя,
    Метаданные.Документы.РасходныйКассовыйОрдер.Синоним);
```

или

Именадокументов = Новый СписокЗначений;
Именадокументов.Добавить(Метаданные.Документы.СтрокаВыпискиРасход.Имя,
 Метаданные.Документы.СтрокаВыпискиРасход.Синоним);
Именадокументов.Добавить(Метаданные.Документы.РасходныйКассовыйОрдер.Имя,
 Метаданные.Документы.РасходныйКассовыйОрдер.Синоним);

5. Сложные логические условия в **Если...ИначеЕсли...КонецЕсли** следует переносить следующим образом:

- каждое элементарное условие нужно начинать с новой строки, если длина строки превышает ограничение в 120 символов;
- логические операторы **И**, **ИЛИ** ставятся в начале строки, а не в конце предыдущей строки;
- все условия предваряются стандартным отступом, либо выравниваются по началу первого условия, без учета логического оператора (для выравнивания выражений относительно первой строки рекомендуется использовать пробелы).

Примеры:

```
Если ( ВидОперации = Перечисления.ВидыОперацийПоступлениеMPЗ.ПоступлениеРозница )
    ИЛИ ( ВидОперации = Перечисления.ВидыОперацийПоступлениеMPЗ.ПоступлениеРозницаКомиссия ) Тогда
        Возврат Истина;
КонецЕсли;
```

```
Если ((СтруктураModуля[Индекс].Блок = Перечисления.ТипыБлоковModулей.ЗаголовокПроцедуры)
    ИЛИ(СтруктураModуля[Индекс].Блок = Перечисления.ТипыБлоковModулей.ЗаголовокФункции) )
    И(Найти(ВРЕГ(СтруктураModуля[Индекс].Текст), КлючБлока)> 0) Тогда
```

6. Для выполнения перечисленных выше рекомендаций, кроме автоматического форматирования текста программного модуля, в процессе ввода можно также отформатировать уже введенный текст. Для этого необходимо выделить блок текста, который требуется отформатировать, и выбрать пункт меню **Текст — Блок — Форматировать**. При этом текстовый редактор проанализирует текст модуля и выполнит его форматирование, при котором содержимое каждой синтаксической конструкции будет сдвинуто вправо на величину табуляции независимо от первоначального расположения строк (лидирующих пробелов). В пустые строки устанавливаются знаки табуляции в соответствии с синтаксической конструкцией.

Для автоматической расстановки переносов строк можно воспользоваться [приложенной обработкой](#).

Использование дублирующего кода

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

#std440

1. Дублированием кода называется способ разработки конфигурации, при котором при создании нового функционала копируются без изменений уже существующие фрагменты кода или целиком процедуры и функции и при этом копируемый функционал по логике приложения должен быть одинаковым.

Дублирование кода создает проблемы для сопровождения разрабатываемой конфигурации:

- в копии попадают все ошибки из дублируемого кода;
- при исправлении ошибок существует вероятность пропустить некоторые вхождения;
- затраты на исправление ошибок увеличиваются;
- усложняется понимание структуры программы.

Дублирование часто возникает из-за невозможности доступа к написанному прежде коду (например, если написанный прежде код размещен в модуле той или иной формы, а его использование востребовано при разработке другой формы).

Следует осторожно относиться к дублированию кода и по возможности стараться его избегать. Основной способ избежать дублирования кода - переработать существующий код. Это позволит вывести процедуры и функции, алгоритмы которых могут быть использованы повторно, из модулей объектов и модулей форм в общие модули.

2. Следует помнить, что дублирование кода оправданно и должно выполняться, если развитие функционала в будущем может привести к значительному расхождению двух вариантов кода.

Пример предотвращения дублирования кода при разработке клиент-серверных функций

В функции **СообщитьПользователю** общего модуля **ОбщегоНазначенияКлиентСервер** возникла необходимость особым образом обработать входные параметры при выполнении на сервере.

Неправильно использовать для этого [инструкция препроцессора](#) (#Если НЕ ТонкийКлиент И НЕ ВебКлиент):

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь)

Экспорт

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = ТекстСообщенияПользователю;
Сообщение.Поле = Поле;
```

ЭтоОбъект = Ложь;

```
#Если НЕ ТонкийКлиент И НЕ ВебКлиент Тогда
    Если КлючДанных <> Неопределено
        И XMLТипЗнч(КлючДанных) <> Неопределено Тогда
            ТипЗначенияСтрокой = XMLТипЗнч(КлючДанных).ИмяТипа;
            ЭтоОбъект = СтрНайти(ТипЗначенияСтрокой, "Объект.") > 0;
        КонецЕсли;
    КонецЕсли;
```

```
Если ЭтоОбъект Тогда
    Сообщение.УстановитьДанные(КлючДанных);
Иначе
    Сообщение.КлючДанных = КлючДанных;
КонецЕсли;
```

```
Сообщение.Сообщить();
Отказ = Истина;
```

КонецПроцедуры

Правильно разделить процедуру на две одноименные в серверном и клиентском модуле с различной реализацией, и для того чтобы избежать дублирования кода, общую реализацию оставить в клиент-серверном общем модуле:

1) серверная функция:

Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь)

Экспорт

```
ЭтоОбъект = Ложь;
Если КлючДанных <> Неопределено
    И XMLТипЗнч(КлючДанных) <> Неопределено Тогда
```

```
    ТипЗначенияСтрокой = XMLТипЗнч(КлючДанных).ИмяТипа;
    ЭтоОбъект = СтрНайти(ТипЗначенияСтрокой, "Объект.") > 0;
```

```

КонецЕсли;
ОбщегоНазначенияСлужебныйКлиентСервер.СообщитьПользователю(ТекстСообщенияПользователю, КлючДанных, Поле, Отказ, ЭтоОбъект);
КонецПроцедуры
2) клиентская функция:
Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных = Неопределено, Знач Поле = "", Отказ = Ложь)
Экспорт
ОбщегоНазначенияСлужебныйКлиентСервер.СообщитьПользователю(ТекстСообщенияПользователю, КлючДанных, Поле, Отказ);
КонецПроцедуры
3) общая служебная клиент-серверная реализация в модуле ОбщегоНазначенияСлужебныйКлиентСервер:
Процедура СообщитьПользователю(Знач ТекстСообщенияПользователю, Знач КлючДанных, Знач Поле, Отказ = Ложь, ЭтоОбъект = Ложь) Экспорт
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = ТекстСообщенияПользователю;
Сообщение.Поле = Поле;
Если ЭтоОбъект Тогда
Сообщение.УстановитьДанные(КлючДанных);
иначе
Сообщение.КлючДанных = КлючДанных;
КонецЕсли;
Сообщение.Сообщить();
Отказ = Истина;
КонецПроцедуры

```

См. также

- [Разработка конфигураций с повторным использованием общего кода и объектов метаданных](#)

Использование директив компиляции и инструкций препроцессора

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

#std439

1. Директивы компиляции:

```

&наКлиенте (&AtClient)
&наСервере (&AtServer)
&наСервереБезКонтекста (&AtServerNoContext)

```

следует применять только в коде модулей управляемых форм и в коде модулей команд. В остальных модулях рекомендуется применять инструкции препроцессору.

В серверных или клиентских общих модулях контекст исполнения очевиден, поэтому смысла в директивах компиляции нет. В общих модулях с признаками клиент и сервер применение директив компиляции затрудняет понимание, какие же процедуры (функции) доступны в конечном итоге.

клиент-серверных общих модулях для проверки клиентского и серверного контекстов (#Если Сервер, #Если Клиент) ввиду невозможности надежного определения контекста исполнения. Процедуры и функции, которые работают по-разному при вызове с клиента и с сервера, следует размещать в общих модулях с постфиксами Клиент и Сервер, а не КлиентСервер.

В противном случае невозможно гарантировать корректную работу клиент-серверных процедур и функций в различных режимах работы платформы 1С:Предприятие.

Например, неправильно:

```

Функция КодОсновногоЙзыка() Экспорт
#Если НЕ ТонкийКлиент И НЕ ВебКлиент Тогда
    Возврат Метаданные.ОсновноЙзык.КодЯзыка;
#Иначе
    Возврат СтандартныеПодсистемыКлиент.ПараметрКлиента("КодОсновногоЙзыка");
#КонецЕсли
КонецФункции

```

также неправильно:

```

Функция КодОсновногоЙзыка() Экспорт
#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда
    Возврат Метаданные.ОсновноЙзык.КодЯзыка;
#Иначе
    Возврат СтандартныеПодсистемыКлиент.ПараметрКлиента("КодОсновногоЙзыка");
#КонецЕсли
КонецФункции

```

Правильно: разделить на две одноименные функции в серверном и клиентском модуле с различной реализацией. В общем случае, когда у них имеется определенная общая часть, одинаковая для клиента и сервера, то для того чтобы [избежать дублирования кода](#), этот общий код (и только его) следует оставить в клиент-серверном общем модуле и вызывать его из клиентской и серверной функций, соответственно. Тем самым надежно достигается различное поведение в клиентском и серверном контекстах без использования инструкций препроцессора.

В то же время, как и в обычных клиентских модулях, допустимо ветвление кода для учета специфики различных режимов работы клиентского приложения: веб-клиент, тонкий или толстый клиент (например, #Если ВебКлиент).

3. Не следует разрывать инструкциями препроцессора и областями отдельные грамматические конструкции, выражения, а также объявления и места вызова процедур и функций.

Например, неправильно:

```

Процедура Пример1()
    а = 1
#Область ИмяОбласти
    + 2;
#КонецОбласти // разрыв выражения
КонецПроцедуры

#Область ИмяОбласти
Процедура Пример2()
    // ...
#КонецОбласти // разрыв процедуры
КонецПроцедуры

```

```
Если <...> Тогда
// ...
#Если ВебКлиент Тогда // разрыв блока Если
Иначе
// ...
#КонецЕсли
КонецЕсли;

Результат = Пример4(Параметр1,
#Если Клиент Тогда
    Параметр2, // некорректный вызов функции
#КонецЕсли
    Параметр3);

Данные ошибки диагностируются автоматически с помощью среды разработки 1С:Enterprise Development Tools (EDT).

Правильно использовать инструкции препроцессора без разрыва конструкций.
```

Определение типа значения переменной

Область применения: управляемое приложение, мобильное приложение, обычное приложение. #std442

Определение типа значения переменной необходимо выполнять путем его сравнения с типом, а не каким-либо другим методом.
Правильно:

```
Если ТипЗнч(Ссылка) = Тип("ДокументСсылка.ПоступлениеТоваровУслуг") Тогда
```

Неправильно:

```
Если Ссылка.Метаданные().Имя = "ПоступлениеТоваровУслуг" Тогда
```

Получение метаданных объектов

Область применения: управляемое приложение, мобильное приложение, обычное приложение. #std445

1. В тех случаях, когда известен тип объекта метаданного (справочник, документ, и т.п.), то получение метаданных объекта конфигурации следует выполнять с помощью метода **Метаданные** этого объекта (или ссылки для объектов ссылочного типа), а не путем обращения к свойству глобального контекста **Метаданные**, так как второй способ существенно более медленный.
Правильно:

```
СправочникОбъект.Метаданные()
```

Неправильно:

```
Метаданные.Справочники[ИмяСправочника]
Метаданные.НайтиПоПолномуИмени("Справочник." + ИмяСправочника)
```

2. В тех случаях, когда тип объекта метаданного заранее неизвестен, рекомендуется воспользоваться методом **НайтиПоТипу**, например:

```
// Получить полное имя объекта метаданных вида "Справочник.Номенклатура", "Документ.ПриходнаяНакладная" по переданной ссылке.
ИмяОбъектаМетаданного = Метаданные.НайтиПоТипу(ТипЗнч(Ссылка)).ПолноеИмя();
```

Обработчики событий модуля формы, подключаемые из кода

Область применения: управляемое приложение, мобильное приложение, обычное приложение. #std492

Обработчикам событий модуля формы, которые устанавливаются из кода с помощью метода **УстановитьДействие**, рекомендуется задавать префикс **Подключаемый_** (англ. **Attachable_**).
Например:

```
Процедура Подключаемый_РазрешитьРедактированиеРеквизитовОбъекта()
...
Процедура Подключаемый_КонтактнаяИнформацияНачалоВыбора()
...
```

В случае когда подключение обработчика выполняется не в тексте модуля формы (а например, в общем модуле), то в результатах проверки конфигурации с включенным флагом "Поиск неиспользуемых процедур и функций" окажутся ошибки вида:

```
Справочник._ДемоПартнеры.Форма.ФормаЕлемента.Форма Не обнаружено ссылок на процедуру: "
Подключаемый_КонтактнаяИнформацияНачалоВыбора"
```

Использование префикса позволяет легко идентифицировать такие обработчики в результатах проверки и отсеивать как исключения.

Если же подключение обработчика выполняется в тексте модуля формы, то проверка конфигурации с включенным флагом "Поиск неиспользуемых процедур и функций" ошибку не регистрирует.

Использование переменных в программных модулях

Область применения: управляемое приложение, мобильное приложение, обычное приложение. #std639

1. В большинстве случаев, вместо переменных программных модулей следует использовать более подходящие средства разработки платформы **1С:Предприятие**. Поскольку область видимости (использования) таких переменных сложно контролировать, то они зачастую становятся источником трудновоспроизводимых ошибок.
Примеры некорректного использования и исключений из этого правила приведены далее. Рекомендации по оформлению переменных в коде программных модулей см. в статье [Структура модуля](#).

2. Неоправданные примеры использования переменных в модулях объектов (справочников, документов, наборов записей, обработок, отчетов и пр.).

2.1. Для передачи параметров между обработчиками подписок на события и в обработчики событий модуля объекта из внешнего кода рекомендуется использовать свойство объекта **ДополнительныеСвойства**. Например, неправильно:

```
Перем КонвертацияФайлов Экспорт;
Процедура ПередЗаписью(Отказ)
    Если КонвертацияФайлов Тогда
```

КонецПроцедуры

```
// вызывающий код  
ФайлОбъект.Конвертацияфайлов = Истина;  
ФайлОбъект.Записать();
```

Правильно:

Процедура ПередЗаписью(Отказ)

```
Если ДополнительныеСвойства.Свойство("Конвертацияфайлов") Тогда
```

...

КонецПроцедуры

```
// вызывающий код
```

```
ФайлОбъект.ДополнительныеСвойства.Вставить("Конвертацияфайлов", Истина);  
ФайлОбъект.Записать();
```

В то же время, для передачи внутренних параметров между обработчиками событий модуля объекта целесообразно использовать неэкспортные переменные модуля объекта, которые недоступны из внешнего кода.

Например:

Перем ПредыдущееЗначениеОрганизации; // значение реквизита "Организация" до записи объекта в базу

Процедура ПередЗаписью(Отказ)

```
ПредыдущееЗначениеОрганизации = ...; // с помощью запроса выясняем значение до записи объекта в базу
```

КонецПроцедуры

Процедура ПриЗаписи(Отказ)

```
Если ПредыдущееЗначениеРеквизита <> Организация Тогда  
    // отрабатываем изменение значения реквизита при записи
```

...

КонецЕсли;

КонецПроцедуры

2.2. Для обработки кодов возврата (ошибок) в логике программного модуля рекомендуется использовать строковые константы.

Например, неправильно:

```
Перем НетОшибок,  
Ошибка_ОбработкиПроверкиЗаполнения, // возникает, если обработка проверки заполнения вернула отказ  
Ошибка_ЗаписиОбъекта, // возникает, если во время записи объекта возникло исключение  
Ошибка_БлокировкиОбъекта, // возникает, при попытке блокировки объекта
```

Процедура ВыполнитьПерерасчет()

```
Результат = ОбработатьДокументы(...);  
Если Результат = Ошибка_ЗаписиОбъекта Тогда  
    ...  
ИначеЕсли Результат = Ошибка_БлокировкиОбъекта Тогда  
    ...  
ИначеЕсли ...
```

КонецПроцедуры

...

```
////////////////////////////////////////////////////////////////////////  
// ИНИЦИАЛИЗАЦИЯ МОДУЛЯ
```

```
НетОшибок = 1;  
Ошибка_ОбработкиПроверкиЗаполнения = 2;  
Ошибка_ЗаписиОбъекта = 3;  
Ошибка_БлокировкиОбъекта = 4;
```

правильно:

Процедура ВыполнитьПерерасчет()

```
Результат = ОбработатьДокументы(...);  
Если Результат = "ОшибкаЗаписиОбъекта" Тогда  
    ...  
ИначеЕсли Результат = "ОшибкаБлокировкиОбъекта" Тогда  
    ...  
ИначеЕсли ...
```

КонецПроцедуры

модули с повторным использованием возвращаемых значений на время вызова сервера.

Исключение из этого правила составляют случаи, когда по соображениям безопасности возвращать результат вычисления в экспортной функции недопустимо. В этом случае они размещаются в локальной переменной модуля.

3. Неоправданные примеры использования переменных в модулях форм.

модули с повторным использованием возвращаемых значений.

При этом не следует кешировать статическую и легко вычисляемую информацию. В частности, не следует кешировать в клиентских переменных модуля формы значения предопределенных элементов и перечислений. Для их получения на клиенте предназначен [метод ПредопределенноеЗначение](#).

3.2. Для хранения и передачи промежуточных результатов вычислений между разными процедурами и функциями формы следует использовать

- [Параметры процедур и функций](#) – для передачи результатов по цепочке вызовов дочерних процедур и функций в контексте одного вызова.
- Реквизиты формы – если требуется сохранять промежуточные результаты между разными вызовами с клиента. (Следует иметь в виду, что значения серверных переменных модуля формы не сохраняются между вызовами с клиента.)

Исключение из этого правила составляют случаи использования клиентских переменных формы для хранения промежуточных результатов в обработчиках ожидания формы, в обработчиках внешних событий и в клиентских обработчиках событий элементов формы.

Например:

&НаКлиенте

```
Перем ПорядковыйНомерИзображения; // счетчик-нумератор для наименования файлов при сканирования нескольких изображений
```

```

...
&На клиенте
Процедура ВнешнееСобытие(Источник, Событие, Данные)
Если Источник = "TWAIN" И Событие = "ImageAcquired" Тогда
    Если ПорядковыйНомерИзображения = Неопределено Тогда
        ПорядковыйНомерИзображения = 1;
    КонецЕсли;
    ПорядковыйНомерИзображения = ПорядковыйНомерИзображения + 1;
    // Сохранение отсканированного документа в файл с номером ПорядковыйНомерИзображения
    // ...
КонецЕсли;
КонецПроцедуры

```

Использование параметров сеанса.

Предварительная инициализация локальных переменных

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

#std494

Методическая рекомендация (полезный совет)

В случаях когда фрагмент кода вычисляет значение одной или нескольких локальных переменных, рекомендуется явно выполнять предварительную инициализацию таких переменных. Это позволит избежать потенциальных ошибок времени выполнения, когда значение переменной оказывается Неопределено, а последующий код рассчитывает на определенный тип значения. Например:

```

Если ЧоТоТам Тогда
    МояПеременная = 10;
ИначеЕсли
    // ряд следующих веток
    ...
КонецЕсли;
... = МояПеременная; // если ЧоТоТам не ИСТИНА, то нужно учитывать, что МояПеременная может быть равна Неопределено

```

Правильно:

```

МояПеременная = 0; // значение по умолчанию
Если ЧоТоТам Тогда
    МояПеременная = 10;
ИначеЕсли
    // ряд следующих веток
    ...
КонецЕсли;
... = МояПеременная; // используем значение переменной, которая всегда имеет числовое значение

```

Данная рекомендация имеет смысл для больших блоков Если/ИначеЕсли/Иначе, внутри которых сложно визуально по тексту контролировать инициализацию переменных.

Использование Журнала регистрации

Область применения: управляемое приложение, обычное приложение.

#std498

1. Журнал регистрации предназначен для хранения событий, возникающих в процессе работы пользователей с информационной базой. При администрировании эту информацию часто необходимо анализировать в различных разрезах для того, чтобы например, узнать какие события происходили в определенный момент времени, какие действия выполнял тот или иной пользователь.

2. Рекомендуется производить запись в Журнал регистрации из встроенного языка в тех случаях, когда администратору необходимо сообщить дополнительную диагностическую информацию о событиях, которые не записываются платформой **1С:Предприятие**. Такая необходимость может возникнуть как при выполнении бизнес-логики, вызываемой при интерактивной работе, так и в фоновых (рекламентных) заданиях. Для удобства анализа Журнала регистрации одна его запись должна соответствовать одному событию, а сами записи должны содержать ряд обязательных атрибутов, в разрезе которых проводится анализ.

2.1. **Строковый идентификатор типа события**. Как правило, список типов событий в конфигурации может быть сколь угодно большим, поэтому типы событий рекомендуется группировать по функциональному признаку: «Название группы событий.Название события». Например, правильно записывать события с типами «Поручения.Уведомление о новых задачах» и «Поручения.Уведомление о зависящих задачах» вместо двух «плоских» типов событий «Уведомление о новых задачах» и «Уведомление о зависящих задачах». Текст типа события – локализуем, при этом всегда задается основной язык конфигурации.

2.2. **Уровень важности события**. Критичные события, требующие повышенного внимания администратора (ошибки бизнес-логики, сбои в программе, и т.п.), записываются в Журнал регистрации с уровнем важности «Ошибка». Потенциальные проблемы и не фатальные ошибки регистрируются как «Предупреждения». Для вывода информационных сообщений об успешном завершении той или иной операции используется уровень важности «Информация». Также возможно применять и более низкий уровень важности – «Примечание».

2.3. **Комментарий**. Содержит текстовую неструктурированную информацию о событии. В случае ошибок в этом поле содержится информация, необходимая для расследования причины проблемы. Не следует помещать в комментарий информацию сразу о нескольких событиях. Например, неправильно записывать одно событие с комментарием вида:

```

[01.01.2010 00:00:01] Начало инициализации обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1
[01.01.2010 00:00:02] Окончание инициализации обмена данными (успешно)
[01.01.2010 00:00:03] Начало процесса обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1
[01.01.2010 00:00:04] Начало записи изменений в файл обмена
[01.01.2010 00:00:05] Окончание записи изменений в файл обмена (успешно)
[01.01.2010 00:00:06] Окончание процесса обмена данными по настройке "Обмен данными выгрузка", номер строки настройки: 1
[01.01.2010 00:00:07] Выполнено, Выгрузка данных, Обработано 1 объектов

```

правильно записать столько событий, сколько их реально произошло.

Текст комментария – локализуем. Для записи в Журнал регистрации информации о возникшем исключении следует использовать конструкцию:

Подробное представление ошибки(ИнформацияОб ошибке())

Пример регистрации дополнительных событий в функциональной подсистеме «Мой механизм»:

```

Попытка
ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоязыка),
    УровеньЖурналаРегистрации.Информация, , ,
    НСтр("ru = 'Начато действие!'"));
ДействиеСвозможной ошибкой(ОбъектДействия);
ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоязыка),
    УровеньЖурналаРегистрации.Информация, , ,
    НСтр("ru = 'Завершено действие!'"));
Исключение
ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие с возможной ошибкой'", КодОсновногоязыка),
    УровеньЖурналаРегистрации.Ошибка, , ,

```

```
НСтр("ru = '"Во время выполнения действия произошла неизвестная ошибка.'") + Символы.ПС +
ПодробноеПредставлениеОшибка(ИнформацияОбоИшибке());
КонецПопытки;
КонецПроцедуры
```

где переменная **КодОсновногоЯзыка** содержит код языка для хранения данных в информационной базе. Подробнее см. [Автогенерированные данные в информационной базе: требования по локализации](#), п. 1.

3. Не следует использовать выборку из журнала регистрации в тех задачах, где критична высокая скорость выполнения выборки. Поскольку при больших объемах журнала регистрации скорость выборки падает пропорционально увеличению его объема.

Рекомендуется заводить отдельный регистр для протоколирования интересующих событий или обращаться к специализированным объектам платформы (например, **МенеджерФоновыхЗаданий** для выборки истории выполнения фоновых заданий).

Эту особенность нужно также учитывать при разработке отчетов по журналу регистрации.

См. также

- [Строки интерфейса в модулях конфигурации: требования по локализации](#)

Перехват исключений в коде

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

#std499

1. В общем случае не рекомендуется перехватывать исключения. В частности не нужно перехватывать исключения только ради выдачи сообщения об ошибке. Необработанное исключение в любом случае будет выдано пользователю в виде сообщения об ошибке (а также будет записано в журнал регистрации для администратора, если исключение возникло на сервере).

2. Тем не менее, необходимость перехвата исключений в коде все же возникает. Например, для того чтобы уточнить текст ошибки, дополнив его прикладной, понятной конечному пользователю, информацией. Однако при этом необходимо фиксировать причину ошибки в журнале регистрации для того, чтобы системный администратор имел возможность выполнить диагностику проблемы и при необходимости передать информацию об ошибке в службу технической поддержки.

При этом рекомендуется записывать в журнал регистрации подробное представление исключения, а краткое представление добавлять в текст сообщения пользователю.

3. Частные случаи некорректного использования и перехвата исключений.

Область применения (уточнение): управляемое приложение, обычное приложение.

3.1. Если имеется некоторая серверная бизнес-логика, которая вызывается с клиента при интерактивной работе пользователя:

```
&НаСервере
Процедура ВыполнитьОперацию()
    // код, приводящий к вызову исключения
    ...
КонецПроцедуры
```

то неправильно маскировать от пользователя и администратора исходную проблему:

```
// на клиенте
Попытка
    ВыполнитьОперацию();
Иключение
    ПоказатьПредупреждение(, НСтр("ru = 'Операция не может быть выполнена.'"));
КонецПопытки;
```

Правильно записывать в журнал регистрации подробное представление исключения, а краткое представление добавлять в текст сообщения пользователю:

```
&НаСервере
Процедура ВыполнитьОперацию()
    Попытка
        // код, приводящий к вызову исключения
        ...
    Иключение
        // Запись события в журнал регистрации для системного администратора.
        ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции!'),
            УровеньЖурналаРегистрации.Ошибка, ,
            ПодробноеПредставлениеОшибка(ИнформацияОбоИшибке()));
        ВызватьИключение;
    КонецПопытки;
КонецПроцедуры
```

и тогда на клиенте:

```
Попытка
    ВыполнитьОперацию();
Иключение
    ТекстСообщения = КраткоеПредставлениеОшибка(ИнформацияОбоИшибке());
    ПоказатьПредупреждение(, НСтр("ru = 'Операция не может быть выполнена по причине:'") + Символы.ПС + ТекстСообщения);
КонецПопытки;
```

Не следует использовать функцию **ОписаниеОшибка**, т.к. она неинформативна для разработчика, потому что не возвращает стек в тексте ошибки.

3.2. Не следует анализировать текст исключений с целью интерпретации причины ошибки. Текст исключения может [меняться в зависимости от локализации](#). В условиях отсутствия штатных средств (например, типизированных исключений), следует выдавать пользователю тексты исключений «как есть». Для понятности, его можно дополнить пояснением возможных причин.

Например:

```
Попытка
    ЗагрузитьФайлИзИнтернета(...);
Иключение
    ТекстСообщения = КраткоеПредставлениеОшибка(ИнформацияОбоИшибке());
    ТекстСообщения = НСтр("ru = 'Не удалось загрузить файл:'") + Символы.ПС + ТекстСообщения + Символы.ПС + НСтр("ru = 'Возможные
причины:
    • Нет подключения к Интернету;
    • На веб-узле возникли неполадки;
    • Брандмауэр или другое промежуточное ПО (антивирусы и т.п.) блокируют попытки программы подключиться к Интернету;
    • Подключение к Интернету выполняется через прокси-сервер, но его параметры не заданы в программе.'");
    ПоказатьПредупреждение(, ТекстСообщения);
КонецПопытки;
```

В тех случаях, когда анализ типов исключений критически важен для корректной работы бизнес-логики, следует отказаться от исключений и использовать коды ошибок (коды возврата). При этом недопустимо использовать числовые коды ошибок, т.к. код становится нечитаемым:

```
КодОшибки = ЗагрузитьФайлИзИнтернета(...);
```

```
Если КодОшибки = 12345 Тогда
```

```
...
```

```
Иначе Если ...
```

правильно применять строковые литералы (например, "Успешно", "НетМестаНаДиске", "Отменено" и т.п.):

```
РезультатЗагрузки = ЗагрузитьФайлИзИнтернета(...);
```

```
Если РезультатЗагрузки = "Успешно" Тогда
```

```
...
```

```
Иначе Если ...
```

Строковые литералы кодов ошибок не [локализуются](#).

Исключение составляют случаи работы с веб-сервисами и другими внешними системами, где коды ошибок не доступны, а результат работы транслируется в вызывающий код прикладной конфигурации в виде исключений.

3.3. Если имеется некоторая клиентская бизнес-логика (код выполняется полностью на клиенте):

```
&НаКлиенте
Процедура СоздатьФайлНадиске()
    // код, приводящий к вызову исключения
    ...
КонецПроцедуры
```

то рекомендуется делать дополнительный серверный вызов для протоколирования неудачного результата операции в журнал регистрации:

```
Попытка
    // клиентский код, приводящий к вызову исключения
    СоздатьФайлНадиске();
Исключение
    ТекстСообщения = КраткоеПредставлениеОшибка(ИнформацияОбОшибка());
    ПоказатьПредупреждение(НСтр("ru = 'Операция не может быть выполнена по причине:'") + Символы.ПС + ТекстСообщения);
    ЗаписатьОшибкаРаботыСФайлами(ПодробноеПредставлениеОшибка(ИнформацияОбОшибка()));
КонецПопытки;
```

```
&НаСервереБезКонтекста
Процедура ЗаписатьОшибкаРаботыСФайлами(...)
    ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),
        УровеньЖурналаРегистрации.Ошибка.,
        ПодробноеПредставлениеОшибка);
КонецПроцедуры
```

3.4. Недопустимо перехватывать любые исключения, бесследно для системного администратора:

```
Попытка
    // код, приводящий к вызову исключения
    ...
Исключение // перехват любых исключений
КонецПопытки;
```

Как правило, подобная конструкция скрывает реальную проблему, которую впоследствии невозможно диагностировать.

Правильно:

```
Попытка
    // код, приводящий к вызову исключения
    ...
Исключение
    // Пояснение причин перехвата всех исключений "незаметно" от пользователя.
    // ...
    // И запись события в журнал регистрации для системного администратора.
    ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),
        УровеньЖурналаРегистрации.Ошибка.,
        ПодробноеПредставлениеОшибка(ИнформацияОбОшибка()));
КонецПопытки;
```

См. также [Доступ к файловой системе из кода конфигурации](#), удаление временных файлов

3.5. Неприемлемо в событиях **При записи**, **Перед Записью**, **Перед Удалением**, **Обработка Проведения**, **Обработка Проверки Заполнения** и т.п. вызывать исключения для выдачи останавливающих предупреждений, которые должен отработать пользователь. Вместо этого следует устанавливать параметр **Отказ** в значение **Истина** и выводить сообщения пользователю. Тем самым пользователь увидит все блокирующие причины остановки сразу, а не по очереди.

Но если пользователь не может отработать останавливающее предупреждение в рамках выполняемой операции или ошибочная ситуация носит исключительный характер и делает бессмысленными все остальные проверки, которые выводят другие останавливающие предупреждения, то следует вызывать исключение.

```
Процедура ПередЗаписью(Отказ)
    Если Не ЗарегистрироватьИзмененияНаУзлахПлановОбмена() Тогда
        ВызватьИсключение НСтр("#anchor_400">Информирование пользователя).
```

3.6. Недопустимо делать проверки наличия у объекта реквизитов, методов, макетов и т.п., используя для этого исключение, т.к. это может привести к сложно диагностируемым ошибкам, а также затрудняет отладку в режиме «Останавливаться по ошибке».

Вместо перехвата исключений в этом случае рекомендуется:

- использовать механизмы работы с метаданными, чтобы явным образом проверять наличие или отсутствие реквизита (макета и т.п.);
- если различия связаны с особенностями встраивания библиотек – описывать особенности явным образом в переопределяемых модулях (см. [Переопределяемые и поставляемые объекты](#));
- пересмотреть логику работы методов, использующих перехват исключений. Например, можно предусмотреть параметры, которые определяются в вызывающем коде и указывают нужно или нет обращаться к какому-либо методу или свойству объекта.

Например, неправильно:

```
Попытка
    КонтекстЭДоСервер.ПолучитьМакет("КомпонентаОбмена");
    ПутьВК = КонтекстЭДоСервер.ПутьОбъекту + ".Макет.КомпонентаОбмена";
Исключение
КонецПопытки;
```

Правильно:

```
МакетКомпонентыОбмена = КонтекстЭДоСервер.Метаданные().Макеты.Найти("КомпонентаОбмена");
Если МакетКомпонентыОбмена <> Неопределено Тогда
    ПутьКМакету = КомпонентаОбмена.ПолноеИмя()
КонецЕсли;
```

Поиск в коллекциях значений

1. При двух и более операциях поиска в объекте **ТаблицаЗначений** с большим количеством строк^(*) рекомендуется:

- Индексировать колонки, по которым выполняется поиск;
- Но только те из них, которые обладают хорошей селективностью (т.е. каждому значению этой колонки должно соответствовать небольшое количество строк). В противном случае, индексирование не даст эффекта, либо он будет отрицательным (потрачено лишнее время на индексирование).

* Примечание: следует ориентироваться на 1000 строк и более, а также учитывать не только размер таблицы, в которой выполняется поиск, но и сколько раз он выполняется. Например, даже если таблица относительно небольшая в 100 строк, но поиск по ней выполняется 100 раз, ее тоже имеет смысл индексировать. В то же время, нет смысла индексировать таблицу из-за только одной единственной операции поиска.

2. Для поиска значений предусмотрены два метода объекта **ТаблицаЗначений**:

- **Найти**
- **НайтиСтроки**

При поиске значения в одной колонке таблицы значений оба метода одинаково эффективно используют индекс, если он был задан (см. п.1). Однако при поиске значения сразу по нескольким (или по всем) колонкам необходимо учитывать следующие ограничения.

2.1. Не следует использовать метод **Найти** для поиска по нескольким колонкам в таблицах значений с большим количеством строк, даже если проиндексированы все колонки, обладающие хорошей селективностью. Это ограничение вызвано тем, что метод **Найти** выполняет поиск с применением индекса только по одному полю.

Например:

```
T3.Индексы.Добавить("Колонка1");
T3.Индексы.Добавить("Колонка2");
... = T3.Найти("найдется все", "Колонка1, Колонка2"); // Индекс НЕ используется!
```

В этом примере, несмотря на наличие индекса для колонок **Колонка1** и **Колонка2**, поиск все равно будет выполняться перебором всех строк в таблице значений (что очень ресурсоемко на больших объемах данных).

2.2. При использовании метода **НайтиСтроки** в таблицах значений с большим количеством строк следует обеспечить, чтобы список полей индекса был точно таким же, как он задан в структуре поиска (порядок полей не важен). В противном случае, индекс не будет задействован, и поиск будет выполняться перебором всех строк в таблице значений (что очень ресурсоемко на больших объемах данных).

Например:

```
T3.Индексы.Добавить("Колонка1"); // Индекс1
T3.Индексы.Добавить("Колонка2"); // Индекс2

... = T3.НайтиСтроки(Новый Структура("Колонка1, Колонка2 ", "Ищу1", "Ищу2")); // Индекс НЕ используется!
... = T3.НайтиСтроки(Новый Структура("Колонка1", "Ищу1")); // OK - используется Индекс1
... = T3.НайтиСтроки(Новый Структура("Колонка2", "Ищу2")); // OK - используется Индекс2
```

Другой пример:

```
T3.Индексы.Добавить("Колонка1,Колонка2");

... = T3.НайтиСтроки(Новый Структура("Колонка1, Колонка2", "Ищу1","Ищу2")); // OK - индекс используется
... = T3.НайтиСтроки(Новый Структура("Колонка2, Колонка1", "Ищу2","Ищу1")); // OK - индекс используется
... = T3.НайтиСтроки(Новый Структура("Колонка1", "Ищу1")); // Индекс НЕ используется!
... = T3.НайтиСтроки(Новый Структура("Колонка2", "Ищу2")); // Индекс НЕ используется!
```

2.3. Аналогичное ограничение действует и для метода **Скопировать** таблицы значений при вызове с параметром **ПараметрыОтбора (Структура)**.

3. В тех случаях, когда для таблицы значений применяется сортировка по колонкам, содержащим ссылочные значения, необходимо учитывать, что при этом для каждой из этих колонок для всех строк таблицы значений системой будет выполнено обращение к информационной базе за представлением этой ссылки.

Поэтому рекомендуется:

- В тех случаях, когда требуется сортировка по наименованию – сразу, на этапе заполнения, добавлять в таблицу дополнительные колонки с представлениями, и сортировку выполнять уже по ним. Если, конечно, это не вызовет аналогичных многократных обращений к информационной базе;
- В остальных случаях – сортировать «по ссылке», а не по представлению. Для этого в методе **Сортировать** следует использовать объект **СравнениеЗначений**:

```
ОбъектСравнения = Новый СравнениеЗначений;
Таблицадокументов.Сортировать("Дата,Ссылка", ОбъектСравнения);
```

Особенно это важно для таблиц с большим количеством (несколько сотен и тысяч) строк, в алгоритмах критических ко времени исполнения.

3.1. При поиске в объекте **Массив** с большим количеством элементов^(*) следует отказаться от массива в пользу:

- объекта **Соответствие**, если не важен порядок элементов;
- индексированной **ТаблицаЗначений**, если порядок элементов значим.

Это обусловлено тем, что в указанных случаях поиск занимает в большинстве случаев константное время, а в массиве поиск выполняется перебором и поэтому пропорционален количеству элементов.

* Примечание: следует ориентироваться на 1000 элементов и более, а также учитывать не только размер массива, но и сколько раз выполняется поиск. Например, если поиск выполняется многократно, в частности, в цикле, то эта рекомендация также действительна для массивов меньшего размера (до 1000 элементов). Особого внимания требуют универсальные механизмы, которые могут применяться на сколь угодно больших объемах данных.

3.2. При необходимости обеспечить уникальность элементов в большом массиве следует однократно в конце алгоритма вызвать функцию **СвернутьМассив** или процедуру **ДополнитьМассив** с параметром **ТолькоУникальныеЗначения** = **Истина** (модуль **ОбщегоНазначения** Библиотеки стандартных подсистем).

4. Аналогичный недостаток существует и у объекта **ДеревоЗначений**, в котором не предусмотрено индексов и поиск выполняется перебором (как в массиве). В указанных выше случаях объект **ДеревоЗначений** следует заменять индексированным объектом **ТаблицаЗначений**.

Использование объекта РегистрСведенийМенеджерЗаписи

1. Чтение записи (набора записей) из регистра сведений без последующей модификации необходимо выполнять запросом.

2. Объект **РегистрСведенийМенеджерЗаписи** следует применять только тогда, когда выполнение операций с регистром сведений требует использования отбора одновременно по всем измерениям. При этом менеджер записи использует для выполнения записи два набора записей, устанавливая им соответствующие значения отборов. Поэтому обработчики событий набора записей вызываются и тогда, когда для записи данных используется менеджер записи.

3. В остальных случаях следует использовать объект **РегистрСведенийНаборЗаписей**. С точки зрения производительности использование менеджера записей в некоторых случаях будет столь же эффективным, как и использование набора записей, а в некоторых - менее, так как будут выполняться лишние действия.

Правильно:

```
Набор = РегистрыСведений.ЗначенияПравПользователя.СоздатьНаборЗаписей();
Набор.Отбор.НаборПрав.Установить(ЗначениеНабораПрав);
Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав Цикл
    Запись = Набор.Добавить();
    Запись.НаборПрав = ЗначениеНабораПрав;
    Запись.Право = СтрокаТаблицы.Право;
    Запись.Значение = СтрокаТаблицы.Значение;
КонецЦикла;
Набор.Записать();
```

Неправильно:

```
Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав Цикл
    ЭлементРегистраСведений = РегистрыСведений.ЗначенияПравПользователя.СоздатьМенеджерЗаписи();
    ЭлементРегистраСведений.НаборПрав = ЗначениеНабораПрав;
    ЭлементРегистраСведений.Право = СтрокаТаблицы.Право;
    ЭлементРегистраСведений.Значение = СтрокаТаблицы.Значение;
    ЭлементРегистраСведений.Записать();
КонецЦикла;
```

Копирование строк между таблицами значений (табличными частями и т.п.) произвольной структуры

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

#std448

При копировании строк между различными таблицами значений (табличными частями и т.п.) со схожим составом колонок следует использовать метод глобального контекста **ЗаполнитьЗначенияСвойств**.

Алгоритмы, использующие данный метод значительно эффективнее, чем например, многократный перебор колонок таблицы значений, выполняемый для получения их состава.

Правильно:

```
Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл
    СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();
    ЗаполнитьЗначенияСвойств(СтрокаТаблицыПриемника, СтрокаТаблицыИсточника);
КонецЦикла;
```

Неправильно:

```
Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл
    СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();
    Для каждого Колонка Из ТаблицаПриемник.Колонки Цикл
        КолонкаТаблицыИсточника = ТаблицаИсточник.Колонки.Найти(Колонка.Имя);
        Если КолонкаТаблицыИсточника <> Неопределено Тогда
            СтрокаТаблицыПриемника[Колонка.Имя] = СтрокаТаблицыИсточника[Колонка.Имя];
        КонецЕсли;
    КонецЦикла;
КонецЦикла;
```

Порядок записи движений документов

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

#std450

1. Не рекомендуется использовать явную запись наборов записей регистров (с помощью метода **Записать**) в процедурах обработки проведения документов. Запись должна производится неявно системой, при завершении процедуры проведения.

В случае же нарушения этого правила, при параллельной работе нескольких пользователей, возможна ситуация возникновения взаимных блокировок при проведении документов.

2. Исключением является ситуация, когда данные, сохраняемые в регистрах, необходимы в последующих алгоритмах, выполняемых до момента выхода из процедуры проведения.

См. также

- [Требования к проведению документов](#)
- [Самодостаточность регистров](#)
- [Использование активности движений](#)

Получение представлений для ссылочных значений в табличном документе

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

#std449

Методическая рекомендация (полезный совет)

При формировании табличного документа запрещено в качестве параметров ячеек с типом заполнения **Параметр** указывать ссылочные значения, поскольку в этом случае в момент вывода данных в табличный документ будет выполнено многократное обращение к базе данных для получения представлений этих значений. Поэтому в качестве параметров следует указывать сами представления.

Исключением могут быть случаи, когда для получения представлений придется выполнять аналогичное многократное обращение к базе данных.

При этом следует иметь ввиду, что при получении представлений для полей непосредственно в самом запросе (через поле **Представление** или функцией **Представление(<Имя поля>)**) выполняется неявное соединение с таблицей объекта, для которого получаются представления. Для полей составного типа – несколько соединений, для каждого из типов, входящих в состав. Это может приводить к увеличению времени выполнения запроса (и как следствие, общего времени формирования итогового документа), а при большом количестве типов – к невозможности его выполнения в клиент-серверной версии из-за ограничения **Microsoft SQL Server 2005**, по которому в запросе не может участвовать больше 256 таблиц. Такие случаи также могут быть исключением для данного правила, в них представления для ссылочных значений допускается получать в момент их вывода в табличный документ.

Поскольку однозначно рекомендовать – какой из способов получения представлений следует выбрать – нельзя, такой выбор должен делаться разработчиком самостоятельно, на основании данных, полученных экспериментально.

См. также

- [Особенности работы с полем Представление и функцией Представление\(\) языка запросов \(статья на ИТС\)](#)
- [Вывод ссылочных полей](#)

Программное создание прикладных объектов

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

#std451

1. Для программного создания прикладных объектов следует использовать методы соответствующих менеджеров (**СоздатьЭлемент**, **СоздатьДокумент**, **СоздатьНаборЗаписей** и т.д.).

Для программного создания прикладных объектов, у которых существует соответствующие менеджеры объектов, использование конструктора (оператор встроенного языка **Новый**) запрещается.

Правильно:

`ДокументПриходная = Документы.ПоступлениеТоваровУслуг.СоздатьДокумент();`

Неправильно:

`ДокументПриходная = Новый("ДокументОбъект.ПоступлениеТоваровУслуг");`

2. При программном создании объекта следует явно вызывать метод объекта **Заполнить**. Если данных для заполнения нет, то передать значение **Неопределено**. В этом случае корректно отработают свойства реквизитов объекта "**Значение заполнения**", будет вызван обработчик **ОбработкаЗаполнения** и подписи на это событие, как при интерактивной работе с объектом.

Например, неправильно:

```
Папка = Справочники.ПапкиФайлов.СоздатьЭлемент();  
// ...  
Папка.Записать();
```

Правильно:

```
Папка = Справочники.ПапкиФайлов.СоздатьЭлемент();  
// ...  
Папка.Заполнить(Неопределено);  
// ...  
Папка.Записать();
```

Исключением могут быть случаи, когда объект полностью загружается из источника при обмене данными или восстановление базы из резервной копии (загрузка из XML).

Использование модуля объекта, модуля менеджера объекта и общих модулей

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

#std486

Методическая рекомендация (полезный совет)

1. **Модуль объекта** предназначен для реализации поведения отдельного экземпляра объекта (**СправочникОбъект**, **ДокументОбъект** и т.п.). В модуле объекта размещаются процедуры и функции, которые работают с данными объекта (**ЭтотОбъект** и переменные модуля объекта), в том числе когда он еще не записан в информационную базу.

Например, в модуле объекта могут размещаться:

- обработчики событий объекта
- процедуры заполнения экземпляра объекта.

Следует иметь в виду, что для вызова экспортных процедур и функций модуля объекта из других модулей может потребоваться предварительно получить сам экземпляр объекта из информационной базы с помощью метода **ПолучитьОбъект**. При этом происходит загрузка объекта из базы целиком, вместе с его табличными частями, что достаточно ресурсоемко.

См. также: [Чтение отдельных реквизитов объекта из базы данных](#)

2. **Модуль менеджера** объекта предназначен для размещения "статической" функциональности, которая логически неразрывно связана с объектом метаданных, но не зависит от состояния конкретного экземпляра объекта данных. Это могут быть процедуры и функции:

- относящиеся не к одному, а сразу к некоторой совокупности объектов. Например, это функции для вывода на печать списка объектов; функции, возвращающие информацию, общую для всех экземпляров объекта метаданных; процедуры обновления данных информационной базы, которые связаны с объектом метаданных; и т.п.
- которые работают с объектом, записанным в ИБ. В таких функциях входным параметром является ссылка на объект. Например, это функции для получения печатной формы по ссылке на объект, процедуры формирования движений по ссылке на объект и т.п.

Для выполнения функций модуля менеджера объекта не должен требоваться экземпляр объекта данных (**СправочникОбъект**, **ДокументОбъект** и т.п.).

3. Если функциональность невозможно однозначно отнести к тому или иному объекту метаданных, то она является логически общей для нескольких объектов. В этом случае ее следует размещать в **общем модуле**.

Ограничения на использование экспортных процедур и функций

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

#std544

Не следует размещать экспортные процедуры и функции в модулях команд и общих командах. К этим модулям нет возможности обращаться из внешнего по отношению к ним кода, поэтому экспортные процедуры и функции в этих модулях не имеют смысла.

Установка параметров выбора и связей параметров выбора для объектов метаданных

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

#std411

1. Как правило, такие ограничения бизнес-логики как ограничения выбора, должны быть одинаковыми для всех форм, в которых редактируется тот или иной объект. Поэтому задавать параметры выбора и связи параметров выбора рекомендуется в свойствах объектов метаданных - в реквизитах справочников, документов и т.п.

2. Однако могут встречаться случаи, когда ограничения выбора могут зависеть от конкретного сценария работы. В таких случаях параметры выбора могут быть уточнены по месту, в конкретной форме.

Например, в конфигурации имеются

- справочник **Сотрудники**, в котором есть реквизиты **Организация** и **ВидСотрудника** (перечисление со значениями **Основной/Совместитель**);
- документ **ПриказОПрием**, в котором есть реквизит **Организация** и **Сотрудник**; при этом для реквизита **Сотрудник** документа **ПриказОПрием** задана связь параметра выбора **Отбор.Организация** с реквизитом **Организация**.

Требуется реализовать возможность выбора только основных сотрудников в зависимости от значения функциональной опции **ВыбратьТолькоИзОсновныхСотрудников**. Для этого необходимо

- в форме документа **ПриказОПриеме** реализовать дополнительный реквизит формы **ВидыСотрудников**,
- после чего для поля формы **Сотрудник** может быть установлена связь параметра выбора **Отбор.ВидСотрудника** с реквизитом формы **ВидыСотрудников**,
- при этом реквизит формы **ВидыСотрудников** заполняется на основании анализа функциональной опции.

(При этом установить для реквизита **Сотрудник** документа **ПриказОПриеме** связь для параметра выбора **Отбор.ВидСотрудника** нет возможности, т.к. реквизита **ВидСотрудника** в документе **ПриказОПриеме** не существует.)

Тогда установка связи для параметра **Отбор.Организация** в свойствах реквизита **Сотрудник** документа **ПриказОПриеме** и связи для параметра **Отбор.ВидСотрудника** в поле формы документа **ПриказОПриеме** приведет к тому, что в режиме **1С:Предприятия** будут работать обе связи параметра выбора. Таким образом, при выборе сотрудника в форме приказа о приеме отбор в списке сотрудников будет установлен как по организации, заполненной в документе, так и по виду сотрудника, который будет определен на основании функциональной опции.

Использование РеквизитФормыВЗначение и ДанныеФормыВЗначение

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

#std409

Методическая рекомендация (полезный совет)

В большинстве случаев, в модулях форм следует использовать метод формы **РеквизитФормыВЗначение** вместо метода **ДанныеФормыВЗначение**.

Рекомендация обусловлена соображениями унификации прикладного кода и тем, что синтаксис метода **РеквизитФормыВЗначение** проще, чем у **ДанныеФормыВЗначение** (а следовательно, меньше вероятность ошибки).

В **ДанныеФормыВЗначение** необходимо дополнительно передавать тип значения:

ТаблицаПодписей = **ДанныеФормыВЗначение**(**ТаблицаПодписей**, Тип("ТаблицаЗначений"));

а для **РеквизитФормыВЗначение** это не обязательно, а в практическом плане - избыточно:

ТаблицаПодписей = **РеквизитФормыВЗначение**("ТаблицаПодписей");

Наличие в платформе **1С:Предприятие** метода формы **РеквизитФормыВЗначение** (наряду с методом глобального контекста **ДанныеФормыВЗначение**) объясняется только удобством его применения. С точки зрения эффективности и результата методы работают одинаково.

Применение параметров отчета в СКД

Область применения: управляемое приложение, обычное приложение.

#std407

При использовании параметров в отчетах, реализуемых с помощью системы компоновки данных, следует придерживаться следующих рекомендаций.

- Избегать использования параметров отчета в том случае, если можно обойтись "обычным" управлением элементами отбора. Например, не применять в запросах СКД условия типа:

ГДЕ Организация = &Организация

Взамен этого дать пользователю возможность управлять элементом отбора **Организация**.

- Избегать применения обязательных параметров для того, чтобы исключить ситуацию, когда пользователь может отключить параметры отчета, ожидая отмены их применения, и получить при этом ошибку выполнения "Не задан параметр отчета". Для этого следует использовать синтаксические элементы расширения языка запросов системы компоновки данных **{ГДЕ...}**.

Например, применение конструкции

{ГДЕ Сведения.Период >= &датаНачала, Сведения.Период <= &датаОкончания}

приведет к тому, что оба параметра отчета **ДатаНачала** и **ДатаОкончания** будут необязательными и при их отключении пользователем будут отключены соответствующие фрагменты условия.

Использование объектов типа Структура

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

#std693

Требования, предъявляемые данным стандартом к **Структурам**, направлены на повышение читаемости кода и упрощение внесения изменений в код разными авторами (разработчиками) как при коллективной разработке, так и при доработке прикладных решений на конкретных внедрениях. Повышение читаемости кода в свою очередь ведет к уменьшению допускаемых при разработке ошибок и повышает качество прикладного решения.

1. При создании объекта типа **Структура** не рекомендуется передавать в конструктор более 3-х значений свойств. Вместо этого рекомендуется использовать метод **Вставить** или присваивать значения свойствам явным образом

Неправильно

```
ПараметрыФормыКомпоновки = Новый Структура(
    "НеПомещатьНастройкиВСхемуКомпоновкиданных",
    |НеРедактироватьСхемуКомпоновкиданных,
    |НенастраиватьУсловноеОформление,
    |НенастраиватьВыбор,
    |НенастраиватьПорядок,
    |АдресСхемыКомпоновкиданных,
    |АдресНастройкиКомпоновкиданных,
    |УникальныйИдентификатор,
    |Заголовок",
    Истина,
    Истина,
    Истина,
    Истина,
    Истина,
    ТекущиеДанные.АдресСхемыКомпоновкиданных,
    ?(АдресНастройкиСхемыКомпоновкиданных <> Неопределено,
        АдресНастройкиСхемыКомпоновкиданных,
        ТекущиеДанные.АдресНастройкиСхемыКомпоновкиданных),
    УникальныйИдентификатор,
    ЗаголовокФормыНастройкиСхемыКомпоновкиданных));
```

Правильно

ПараметрыФормыКомпоновки = Новый Структура;

ПараметрыФормыКомпоновки.Вставить("НеПомещатьНастройкиВСхемуКомпоновкиданных", Истина);
ПараметрыФормыКомпоновки.Вставить("НеРедактироватьСхемуКомпоновкиданных", Истина);

ПараметрыФормыКомпоновки.Вставить("НеРедактироватьСхемуКомпоновкиДанных", Истина);
ПараметрыФормыКомпоновки.Вставить("НеНастраиватьВыбор", Истина);
ПараметрыФормыКомпоновки.Вставить("НеНастраиватьПорядок", Истина);
ПараметрыФормыКомпоновки.Вставить("АдресСхемыКомпоновкиДанных", ТекущиеДанные.АдресСхемыКомпоновкиДанных);
ПараметрыФормыКомпоновки.Вставить("АдресНастроекКомпоновкиДанных", ?(АдресНастроекСхемыКомпоновкиДанных <> Неопределено,

АдресНастроекСхемыКомпоновкиДанных,

 ТекущиеДанные.АдресНастроекСхемыКомпоновкиДанных));

ПараметрыФормыКомпоновки.Вставить("УникальныйИдентификатор ", УникальныйИдентификатор);
ПараметрыФормыКомпоновки.Вставить("Заголовок", ЗаголовокФормыНастройкиСхемыКомпоновкиДанных);

2. Не рекомендуется в конструкторе структуры использовать конструкторы других объектов, если эти конструкторы принимают параметры. В частности в конструкторе одной структуры не рекомендуется создавать другие структуры с объявлением значений свойств.

Неправильно

НоменклатураСервер.ЗаполнитьСлужебныеРеквизитыПоНоменклатуреВКоллекции(

 Объект.Товары,
 Новый Структура(
 "ЗаполнитьПризнакХарактеристикиИспользуются",
 |ЗаполнитьПризнакТипНоменклатуры, ЗаполнитьПризнакВариантОформленияПродажи",
 Новый Структура("Номенклатура", "ХарактеристикиИспользуются"),
 Новый Структура("Номенклатура", "ТипНоменклатуры"),
 Новый Структура("Номенклатура", "ВариантОформленияПродажи")
)
);

Правильно

ПараметрыЗаполненияРеквизитов = Новый Структура;
ПараметрыЗаполненияРеквизитов.Вставить("ЗаполнитьПризнакХарактеристикиИспользуются",
 Новый Структура("Номенклатура", "ХарактеристикиИспользуются"));
ПараметрыЗаполненияРеквизитов.Вставить("ЗаполнитьПризнакТипНоменклатуры",
 Новый Структура("Номенклатура", "ТипНоменклатуры"));
НоменклатураСервер.ЗаполнитьСлужебныеРеквизитыПоНоменклатуреВКоллекции(Объект.Товары,
 ПараметрыЗаполненияРеквизитов);

3. Не рекомендуется в конструкторе структуры вызывать функции с большим (более 3) количеством параметров.

Неправильно

СведенияяОтоваре = Новый Структура("ПараметрыТовара, ЦенаПродажиИОстаткиТовара, ЦенаЗакупкиИОстаткиТовара",
 ПодборТоваровКлиентСервер.ПараметрыТовара(),
 ПодборТоваровВызовСервера.ЦенаПродажиИОстаткиТовара(
 Номенклатура,
 Характеристика,
 Соглашение,
 Валюта,
 ВидыЦен),
 ЦенаЗакупкиИОстаткиТовара(
 Номенклатура,
 Характеристика,
 Соглашение,
 Валюта,
 ВидыЦен));

Правильно

СведенияяОтоваре = Новый Структура("ПараметрыТовара, ЦенаПродажиИОстаткиТовара, ЦенаЗакупкиИОстаткиТовара");
СведенияяОтовара.ПараметрыТовара = ПодборТоваровКлиентСервер.ПараметрыТовара();
СведенияяОтовара.ЦенаПродажиИОстаткиТовара = ПодборТоваровВызовСервера.ЦенаПродажиИОстаткиТовара(
 Номенклатура,
 Характеристика,
 Соглашение,
 Валюта,
 ВидыЦен);
СведенияяОтовара.ЦенаПродажиИОстаткиТовара = ЦенаЗакупкиИОстаткиТовара.ЦенаЗакупкиИОстаткиТовара(
 Номенклатура,
 Характеристика,
 Соглашение,
 Валюта,
 ВидыЦен);

См. также

- [Параметры процедур и функций](#)
- [Особенности использования структур в качестве параметров процедур и функций](#)

Особенности сортировки в таблице значений

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

#std781

В тех случаях, когда для таблицы значений применяется сортировка по колонкам, содержащим ссылочные значения, необходимо учитывать, что при этом для каждой из этих колонок для всех строк таблицы значений системой будет выполнено обращение к информационной базе за представлением этой ссылки.

Поэтому рекомендуется:

- В тех случаях, когда требуется сортировка по наименованию – сразу, на этапе заполнения, добавлять в таблицу дополнительные колонки с представлениями, и сортировку выполнять уже по ним. Если, конечно, это не вызовет аналогичных многократных обращений к информационной базе;
- В остальных случаях – сортировать «по ссылке», а не по представлению. Для этого в методе **Сортировать** следует использовать объект **СравнениеЗначений**:

ОбъектСравнения = Новый СравнениеЗначений;
Таблицадокументов.Сортировать("Дата, Ссылка", ОбъектСравнения);

Особенно это важно для таблиц с большим количеством (несколько сотен и тысяч) строк, в алгоритмах критических ко времени исполнения.

См. также

- [Упорядочивание результатов запроса](#)

Массовая конкатенация строк

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

#std782

При массовых(*) операциях конкатенации строк следует использовать методы платформы СтрРазделить и СтрСоединить.
Например, неправильно:

```
Для НомерКолонки = 1 По Макет.ШиринаТаблицы Цикл
    ИзвлеченныйТекст = ИзвлеченныйТекст + Символы.ПС + ТекстОбласти;
    ...

```

Правильно:

```
ИзвлеченныеТексты = Новый Массив;
Для НомерКолонки = 1 По Макет.ШиринаТаблицы Цикл
    ИзвлеченныеТексты.Добавить(ТекстОбласти);
    ...
ИзвлеченныйТекст = СтрСоединить(ИзвлеченныеТексты, Символы.ПС);
```

Такая обработка данных не только быстрее выполняется, но и приводит к снижению потребления оперативной памяти.

* Примечание: следует ориентироваться на 1000 операций конкатенации строк и более (эта величина также может быть еще меньше при увеличении длин строк: чем строки длиннее, тем операции выполняются дольше). Особого внимания требует конкатенация в циклах и в универсальных механизмах, которые могут применяться на сколь угодно больших объемах данных. В то же время, не следует отказываться от конкатенации строк в остальных случаях, так как это заметно снижает читаемость кода.

См. также

- [Оформление текстов запросов](#) (раздел "Конкатенация нескольких текстов запросов в пакет")

Использование модулей с повторным использованием возвращаемых значений

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

#std724

1. Общие модули с повторным использованием возвращаемых значений (далее: кэш) предусмотрены для кэширования результатов работы функций, которые в них размещены - на время сеанса или на время вызова. Их следует применять для экономии вычислительных ресурсов сервера и для [минимизации клиент-серверного взаимодействия](#).

См. также: раздел "[Повторное использование возвращаемых значений](#)" документации по платформе 1С:Предприятие, [Использование значений, влияющих на поведение клиентского приложения](#)

2. В то же время, чрезмерное (неправданное) применение общих модулей с повторным использованием возвращаемых значений может приводить к [излишнему потреблению памяти](#).

2.1. Недопустимо создать общие модули с повторным использованием, из которых возвращаются данные, вычисление которых выполняется быстрее, чем получение из кэша. Например, строковые константы. Кроме того, что получение строковой константы каждый раз будет работать гораздо быстрее, чем получение ее из общего модуля с повторным использованием, эти данные будут занимать память кэша.

Например, неправильно размещать в модуле с повторным использованием:

```
Функция ИмяПакетаУправления() Экспорт
    Возврат "ManagementPackage";
КонецФункции
```

Имеет смысл кэшировать данные, полученные из базы данных, внешних источников данных или путем сложных (ресурсоемких) вычислений. Причем в ряде случаев, даже значения, полученные из базы данных, не стоит кэшировать, если выгода от их кэширования – неочевидна. Например, не стоит кэшировать [константы \(объект метаданных\)](#) примитивных типов, поскольку часто они привносят лишь незначительную долю от общего времени выполнения ресурсоемкой операции.

2.2. Следует помещать в кэш только такие данные, к которым потом будут часто обращаться.

В частности, следует иметь в виду, что кэш не хранит данные вечно. Закэшированное значение будет удалено из кэша через 20 минут после вычисления или через 6 минут после последнего использования (в зависимости от того, что наступит раньше*). Кроме этого значение будет удалено при нехватке оперативной памяти в рабочем процессе сервера, при перезапуске рабочего процесса и при переключении клиента на другой рабочий процесс. Поэтому если никто "не успел" воспользоваться данными из кэша, то этот ресурс был потрачен зря.

* Примечание: конкретные цифры могут варьироваться в зависимости от используемой версии платформы 1С:Предприятие.

2.3. Диапазон значений входных параметров функций, размещенных в общих модулях с повторным использованием, не должен быть широким.

Например, в конфигурации предусмотрена функция, получающая на вход контрагента. Если контрагентов в базе очень много, а сценарий работы пользователей таков, что вероятность того, что кто-то за 5 минут обратится к этому же контрагенту, очень невысокая, то ресурсы будут потрачены впустую. Кроме того, если эту «трату» умножить на количество одновременно работающих пользователей, то бесполезные расходы ресурсов становятся значительными.

3. Не следует изменять данные, полученные из кэша. В противном случае, возможны скрытые ошибки в работе программы, а также бесполезное расходование памяти (ресурсов кэша). Поэтому в качестве возвращаемых значений рекомендуется использовать значения, состояние которых изменить нельзя, например: [ФиксированныйМассив](#), [ФиксированнаяСтруктура](#).

Это ограничение вызвано тем, что кэш возвращает каждый раз не копию объекта, а ссылку на один и тот же объект в памяти. Например, если в массив, который возвращает функция с повторным использованием, при каждом вызове при проведении документов дописывать новое значение, то в результате кэш очень быстро «распухнет». Кроме того, при очередном сбросе кэша, добавленные значения будут потеряны и код, который на них опирался, будет работать некорректно.

4. Если в модуле с повторным использованием размещено несколько экспортных функций, которые не только вызываются «снаружи», но и вызывают друг друга, то следует иметь в виду, что результат «внутренних» вызовов не кэшируется.

Например, если в модуле [ОбменДаннымиПовтИсп](#) размещено две экспортных функции:

```
Функция АвтономнаяРаботаПоддерживается() Экспорт
    Возврат ...
КонецФункции
```

```
Функция ЭтоАвтономноеРабочееМесто() Экспорт
    Возврат АвтономнаяРаботаПоддерживается() И ...
КонецФункции
```

которые последовательно вызываются из прикладного кода,

```
... = ОбменДаннымиПовтИсп.АвтономнаяРаботаПоддерживается();
... = ОбменДаннымиПовтИсп.ЭтоАвтономноеРабочееМесто();
```

то функция [АвтономнаяРаботаПоддерживается](#) будет вычислена дважды.

Для того чтобы ее возвращаемое значение всегда получалось из кэша, следует явно указывать имя модуля:

```
Функция ЭтоАвтономноеРабочееМесто() Экспорт
    Возврат ОбменДаннымиПовтИсп.АвтономнаяРаботаПоддерживается() И ...;
```

Конецфункции

5. Если у общего модуля свойство "Повторное использование возвращаемых значений" установлено в значение "На время сеанса", то в значениях, возвращаемых функциями такого модуля, нельзя использовать значения типа **МенеджерВременныхТаблиц**, **Запрос**, объекты базы данных (например, **ДокументОбъект**, **ОтчетОбъект**) причем, как непосредственно, так и в составе любых коллекций. Ограничение вызвано тем, что значения этих типов допустимо использовать только в том же серверном вызове, в котором они были получены (созданы).

Возврат значений этих типов в указанных функциях не проверяется платформой и приводит к трудно диагностируемой остановке работы программы (записи есть только в технологическом журнале).

Данное ограничение распространяется на использование значений во временном хранилище.

Использование значений, влияющих на поведение клиентского приложения

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

#std459

В том случае, если поведение целого ряда форм или команд зависит от одних и тех же значений (параметров, задаваемых пользователем или иных настроек поведения клиентского приложения), для **минимизации клиент-серверного взаимодействия** рекомендуется применять **общие модули с повторным использованием возвращаемых значений** (свойство «Повторное использование возвращаемых значений» равно «На время сеанса»). Применение таких общих модулей допускается, если изменение соответствующего значения в течение сеанса не является критичным, т.е. в течение всего сеанса может использоваться однажды полученное значение.

При этом функции такого общего модуля должны за один вызов возвращать сразу все значения, которые могут понадобиться в тех или иных обстоятельствах.

Например, если при работе всех форм подсистемы регистрации занятости сотрудников, требуются пользовательские параметры «Время занятости по умолчанию», «Время начала рабочего дня» и «Время окончания рабочего дня», то все эти параметры необходимо получать одним вызовом, возвращающим структуру с тремя свойствами:

Настройки = НастройкиПодсистемыРегистрации();
ВремяНачала = Настройки.ВремяНачалаРабочегоДня;
ВремяОкончания = Настройки.ВремяОкончанияРабочегоДня;

Следует помнить, что применение клиентских общих модулей с повторным использованием возвращаемых значений должно быть разумно ограничено. В частности, данные, влияющие на поведение отдельных форм, рекомендуется размещать в реквизитах самой формы, получая их при создании формы на сервере. И только если от некоторого значения (или нескольких значений) зависит поведение команд или большого числа форм приложения, следует применять общие модули с повторным использованием возвращаемых значений.

Не рекомендуется использовать переменные модуля управляемого приложения и модуля обычного приложения для минимизации клиент-серверного взаимодействия.

Получение предопределенных значений на клиенте

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

#std443

Для получения значения ссылок на предопределенные элементы справочников, планов видов характеристик, планов счетов, планов видов расчета, ссылки на значения перечислений, системных перечислений и точек маршрута бизнес-процессов в коде, выполняемом в клиентском коде (тонкий клиент, веб-клиент), где объекты типа **СправочникМенеджер**, <ИмяСправочника>, **ПеречислениеМенеджер**.<ИмяПеречисления> и т.п. не доступны, предназначена функция глобального контекста **ПредопределеноеЗначение**.

Например:

ЮрФизЛицо = ПредопределеноеЗначение("Перечисление.ЮридическоеФизическоеЛицо.ЮридическоеЛицо");

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. При этом не следует реализовывать дополнительные механизмы кеширования на клиенте предопределенных значений. Указанные выше функции не ухудшают **клиент-серверное взаимодействие**: серверный вызов выполняется только при первом обращении к значению, а результат автоматически кешируется.

См. также

- [Использование предопределенных элементов](#)
- Раздел [Работа с предопределенными значениями](#) в документации к платформе 1С:Предприятие (на ИТС)

Минимизация количества серверных вызовов

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

#std487

1.1. Разработку управляемого приложения необходимо вести с контролем количества вызовов серверных процедур и функций из клиентского кода (серверных вызовов), а в некоторых случаях – также объем передаваемых данных между клиентом и сервером (трафик).

Общее количество серверных вызовов складывается из

- обращений на сервер, которые выполняет платформа **1С:Предприятие**,
- и вызовов, которые выполняются из клиентского кода конфигурации – отключение, привносимое конфигурацией по отношению к платформе.

В общем виде, при проектировании клиент-серверного взаимодействия в конфигурации следует руководствоваться тем, что на каждое действие пользователя в клиентском коде конфигурации не должно выполняться дополнительных вызовов на сервер. Любые исключения из этого правила требуют дополнительного обоснования.

1.2. Отдельно для режима с низкой скоростью соединения следует оптимизировать не только количество вызовов, но и объем передаваемых данных между клиентом и сервером (трафик). Отладку клиент-серверного взаимодействия в этом режиме работы рекомендуется проводить в режиме имитации задержек серверных вызовов.

Ниже рассмотрены типовые действия пользователя и даны рекомендации по организации клиент-серверного взаимодействия.

Запуск клиентского приложения

2.1. В простейшем случае, код конфигурации, выполняемый при запуске клиентского приложения, не должен приводить к обращениям на сервер. В тех случаях, когда все же необходимо получать данные с сервера:

- не следует напрямую вызывать серверные процедуры и функции из кода модуля приложения, модуля управляемого приложения и модуля внешнего соединения.
- правильно: за один вызов на сервер передавать на клиент все параметры, необходимые для запуска приложения.

В случае если параметры для запуска клиентского приложения требуется запрашивать с сервера из разных мест клиентского кода, следует разместить такую функцию в общем серверном **модуле с повторным использованием возвращаемых значений**. При первом вызове этой функции происходит одно обращение к серверу, после чего полученное значение автоматически кешируется платформой на клиенте для всех повторных вызовов этой функции.

Пример:

// Фрагмент общего серверного модуля СтандартныеПодсистемыПереопределяемый с повторным использованием возвращаемых значений
Функция ПараметрыРаботыКлиента()

```
Параметры = Новый Структура();
Параметры.Вставить("ИнформационнаяБазаФайловая", ОбщегоНазначения.ИнформационнаяБазаФайловая());
// Инициализация других параметров, необходимых на клиенте при запуске приложения
// Параметры.Вставить(имя параметра, значение параметра);
Возврат Параметры;
Конецфункции
```

Пример клиентского кода, использующего функцию **ПараметрыРаботыКлиента**:

```
ИнформационнаяБазаФайловая = СтандартныеПодсистемыПереопределляемый. ПараметрыРаботыКлиента().ИнформационнаяБазаФайловая;
Если ИнформационнаяБазаФайловая Тогда
// обработка этого случая в клиентском коде
//...
```

См. также: [Использование значений, влияющих на поведение клиентского приложения](#)

Открытие управляемой формы

3.1. В случае если открытие формы выполняется из кода, следует открывать форму за один вызов с помощью метода глобального контекста **ОткрытьФорму** (при использовании версии платформы 1С:Предприятие 8.2 и более ранних версий - также **ОткрытьФормуМодально**). Для передачи параметров в форму следует использовать параметр этих методов **Параметры**.

3.2. При открытии формы не допускается выполнять обращений к серверу из кода модуля формы в обработчиках клиентских событий формы, таких как **ПриОткрытии** и **ПриПовторномОткрытии**. При необходимости обращения из них к серверным данным, следует размещать эти данные в реквизитах формы в **ПриСозданииНаСервере**.

Например, неправильно:

```
НастройкаПроксиСервера = СерверныйМодуль.НастройкаПроксиСервера();
ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера", Новый Структура("Настройка", НастройкаПроксиСервера));
```

правильно:

```
ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера");
```

при этом получение значения константы выполняется в обработчике **ПриСозданииНаСервере** формы **ПараметрыПроксиСервера**.

Выполнение локальной команды управляемой формы

4.1. Выполнение локальной команды формы должно приводить не более чем к одному вызову сервера.

- Если команда выполняет только клиентские операции (приводит к открытию новой формы, устанавливает отбор в списке, меняет стиль оформление и пр.), то все необходимые данные для ее выполнения должны быть заранее переданы на клиент. Рекомендуется заранее готовить эти данные в обработчике события формы **ПриСозданииНаСервере** и размещать их в реквизитах формы.

Пример:

При выборе товаров из списка номенклатуры требуется запретить для пользователя выбор групп номенклатуры: при попытке выбрать группу программа должна выдавать останавливающее сообщение. Для проверки, является ли выбранный элемент номенклатуры группой, неправильно вызывать отдельную серверную функцию. Следует добавить в таблицу значений, которая связана с полем списка на форме, реквизит **ЭтоГруппа** и заполнять его в обработчике события формы **ПриСозданииНаСервере**. Тогда проверка на клиенте выполняется без дополнительного серверного вызова и имеет вид:

```
ТекущаяСтрока = Элемент.ТекущиеДанные;
Если ТекущаяСтрока.ЭтоГруппа Тогда
    Сообщение = Новый СообщениеПользователю();
    Сообщение.Текст = НСтр("ru = 'Выбор группы запрещен.'");
    Сообщение.Сообщить();
    Возврат;
КонецЕсли;
```

- Если команда связана с выполнением бизнес-логики, которую возможно отработать только на сервере, то вся она должна выполняться за один серверный вызов.

Выбор из справочника

4.2. В общем случае, при выборе из справочника допускается выполнять только один серверный вызов из кода, к которому приводят вызов метода глобального контекста **ОткрытьФорму** (или **ОткрытьФормуМодально**).

4.3. В случае если после выбора из справочника необходимо выполнить бизнес-логику, которую возможно отработать только на сервере, то допустимо выполнять ее за один дополнительный серверный вызов.

Выполнение глобальной команды

5.1. При выполнении глобальной команды допускается выполнять только один серверный вызов из кода. В случае если команда открывает форму, то этот вызов должен выполняться при вызове метода глобального контекста **ОткрытьФорму** (или **ОткрытьФормуМодально**).

Выполнение команды формирования отчета

6.1. При выполнении команды формирования отчета не допускается выполнять дополнительных серверных вызовов из кода конфигурации.

В частности, при открытии формы отчета не допускается выполнять обращений к серверу из кода модуля формы в обработчиках клиентских событий формы, таких как **ПриОткрытии** и **ПриПовторномОткрытии**.

Например, неправильно выполнять формирование отчета, использующего систему компоновки данных, из обработчика формы **ПриОткрытии**:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
    ВывестиОтчет();
КонецПроцедуры
```

```
&НаСервере
Процедура ВывестиОтчет()
    // код по формированию отчета...
КонецПроцедуры
```

правильно:

- открывать форму отчета с параметром **СформироватьПриОткрытии = Истина**, либо
- установить параметр формы **СформироватьПриОткрытии** в значение **Истина** в обработчике **ПриСозданииНаСервере**.

Выполнение подбора элементов

7.1. Особенность клиент-серверного взаимодействия при выполнении подбора элементов состоит в необходимости передавать список выбранных элементов между формой объекта и формой подбора. При этом объем передаваемых данных может быть достаточно большим.

В этом случае не рекомендуется передавать потенциально большой массив данных в качестве параметра формы подбора. Потенциально большой массив данных, хранимый в форме подбора в параметре типа **ДанныеФормыКоллекция**, может оказаться на клиенте не полным за счет оптимизации работы управляемой формы. Как результат – для передачи параметра будет выполнено дополнительное "дочитывание" данных формы с сервера.

7.2. В целях оптимизации передачи данных между формой объекта и формой подбора рекомендуется использовать временное хранилище, чтение и запись которого должна выполняться на сервере.

Проиллюстрируем методику использования формы подбора на примере подбора элементов справочников **Товары** в табличную часть **Товары** документа **РасходТовара**. (Из демонстрационной конфигурации по платформе 1С:Предприятие).

Открытие формы подбора из клиентского кода должно приводить не более чем к двум обращениям на сервер. С этой целью локальная команда открытия формы подбора в модуле формы документа **РасходТовара** помещает список товаров из табличной части во временное хранилище (первый вызов) и открывает форму подбора (второй вызов), передавая адрес временного хранилища:

```
// в форме документа
&НаКлиенте
Процедура ОбработчикКомандыПодбора()
    АдресТоваровВХранилище = ПоместитьТоварыВХранилище();
    ПараметрыПодбора = Новый Структура();
    ПараметрыПодбора.Вставить("АдресТоваровДокумента", АдресТоваровВХранилище);
    ПараметрыПодбора.Вставить("ВидЦен ", Объект.ВидЦен);
    ПараметрыПодбора.Вставить("Склад ", Объект.Склад);

    ФормаПодбора = ОткрытьФорму("Документ.РасходТовара.Форма.ФормаПодбора", ПараметрыПодбора, Элементы.Товары );
КонецПроцедуры
```

```
// Функция помещает список товаров во временное хранилище и возвращает адрес
&НаСервере
Функция ПоместитьТоварыВХранилище()
    Возврат ПоместитьВоВременноеХранилище(Объект.Товары.Выгрузить(), "Товар, Цена, Количество"), УникальныйИдентификатор;
КонецФункции
```

Форма подбора получает список выбранных товаров из временного хранилища в обработчике **ПриСозданииНаСервере**:

```
// в форме подбора
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка) Экспорт
    // Сохраним адрес временного хранилища, чтобы поместить в него обратно подобранные товары
    АдресТоваровДокумента= Параметры.АдресТоваровДокумента;
    Объект.Товары.Загрузить(ПолучитьИзВременногоХранилища(АдресТовара));
КонецПроцедуры
```

Закрытие формы подбора должно приводить не более чем к двум обращениям на сервер. При закрытии форма подбора помещает список выбранных товаров во временное хранилище (первый вызов):

```
&НаКлиенте
Процедура ОКВыполнить()
    АдресТовараеВВременномХранилище = ЗаписатьПодборВХранилище();
    ОповеститьОВыборе(АдресТовараеВВременномХранилище);
КонецПроцедуры
```

```
&НаСервере
Функция ЗаписатьПодборВХранилище()
    Возврат ПоместитьВоВременноеХранилище(Товары.Выгрузить(), АдресТоваровДокумента);
КонецФункции
```

Затем форма документа восстанавливает список товаров из временного хранилища (второй вызов на сервер):

```
// Обработка выбора таблицы формы Товары
&НаКлиенте
Процедура ПриОбработкеПодобранныхТоваров(Элемент, АдресТоваровВХранилище, СтандартнаяОбработка) Экспорт
    Если АдресТоваровВХранилище = Неопределено Тогда
        Возврат;
    КонецЕсли;
    ПолучитьТоварыИзХранилища(АдресТоваровВХранилище);
КонецПроцедуры
```

```
&НаСервере
Процедура ПолучитьТоварыИзХранилища(АдресТоваровВХранилище)
    ПодобранныеТовары = ПолучитьИзВременногоХранилища(АдресТоваровВХранилище);
    Объект.Товары.Загрузить(ПодобранныеТовары);
    УдалитьИзВременногоХранилища(АдресТоваровДокумента); // очищается временное хранилище для минимизации расхода оперативной памяти
КонецПроцедуры
```

7.3. При помещении данных во временное хранилище следует выбрать один из двух вариантов:

- помещать данные во временное хранилище на время жизни формы, используя уникальный идентификатор формы и очищать это временное хранилище после использования (см. пример в п. 7.2).
- предварительно выполнять инициализацию временного хранилища и переиспользовать его

В противном случае при многократном повторении действия в форме, например, при многократном подборе товаров, это приводит к излишнему расходу оперативной памяти.

Рассмотрим пример предварительной инициализации временного хранилища для переиспользования:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ)
    АдресТоваров = ПоместитьВоВременноеХранилище(Неопределено, УникальныйИдентификатор); // Инициализируется реквизит формы
КонецПроцедуры
```

```
&НаСервере
Функция ТоварыВоВременномХранилище()
    Возврат ПоместитьВоВременноеХранилище(Товары.Выгрузить(), АдресТоваров);
КонецФункции
```

При переиспользовании временного хранилища не требуется удалять значение из временного хранилища:

```
// в форме документа
&НаСервере
Процедура ПолучитьТоварыИзХранилища(АдресТоваровВХранилище)
    ПодобранныеТовары = ПолучитьИзВременногоХранилища(АдресТоваровВХранилище);
    Объект.Товары.Загрузить(ПодобранныеТовары);
КонецПроцедуры
```

См. также

- [Использование значений, влияющих на поведение клиентского приложения](#)
- [Использование объекта ДанныеФормыКоллекция](#)
- [Оптимизация клиент-серверного взаимодействия прикладных решений \(ИТС\)](#)

Минимизация кода, выполняемого на клиенте

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

#std629

1.1. Необходимо минимизировать объем кода, который выполняется на стороне клиентского приложения. В частности не следует выполнять на клиенте сложные алгоритмы, требующие значительных ресурсов компьютера. В таких случаях выполнение алгоритма на клиенте может занимать гораздо больше времени, чем передача управления с клиента на сервер, выполнение алгоритма на сервере и возврат результата обратно на клиент.

Следует размещать такие алгоритмы в серверном коде, выполняя к ним минимально необходимое число обращений с клиентом.

См. также: [Минимизация количества серверных вызовов](#)

Это требование продиктовано тем, что

- как правило, клиентский компьютер менее производительный, чем серверный компьютер;
- необходимостью приемлемого качества работы в веб-клиенте. Клиентский код выполняется интерпретатором встроенного языка, который в веб-клиенте работает заметно медленнее, чем в тонком или толстом клиенте.

1.2. Рекомендуется оставлять на клиенте такие алгоритмы, скорость работы которых заведомо быстрее, чем затраты, необходимые на вызов одной серверной функции. Например, перерасчет доступности элементов управления в форме при изменении пользователем данных выполняется на клиенте, т.к. [контекстный серверный вызов](#) для сложной формы может сделать работу пользователя в этой форме неприемлемой.

См. также: [Особенности табличного документа в веб-клиенте](#)

2.1. Исключение из этого правила составляют отдельные случаи, когда функциональная подсистема предназначена для работы с программным обеспечением, установленным на клиентском компьютере. Например, работа с торговым оборудованием, интеграция с клиент-банком, формирование печатных форм в офисные программы и т.п.

2.2. В тех случаях когда функциональная подсистема предназначена для работы с клиентским программным обеспечением только в определенных режимах работы клиента, следует использовать директивы препроцессора. Например, для кода, недоступного в веб-клиенте:

```
#Если ВебКлиент Тогда
Предупреждение(НСтр("ru = 'Загрузка адресного классификатора не доступна в веб-клиенте.'"));
#Иначе
ОткрытьФорму("РегистрСведений.АдресныйКлассификатор.Форма.ФормаЗагрузкиАдресногоКлассификатора");
#КонецЕсли
```

Доступ к файловой системе из кода конфигурации

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

#std542

1. При обращении из кода конфигурации к файлам и каталогам файловой системы следует иметь в виду, что доступ к ним ограничен правами пользователя операционной системы, от имени которого запущено приложение.

1.1. Если доступ к файловой системе осуществляется из кода, выполняемого на клиенте, то он выполняется под правами пользователя, от имени которого запущено приложение (тонкий, толстый или веб-клиент). Как правило, это текущий пользователь операционной системы.

1.2. Если доступ к файловой системе осуществляется из кода, выполняемого на сервере, то:

- при использовании клиент-серверной информационной базы, доступ ограничен правами пользователя, от имени которого запущен сервер 1С:Предприятия (*);
- при использовании файловой базы, опубликованной на веб-сервере, доступ ограничен правами пользователя, от имени которого запущен веб-сервер.

* Рабочие процессы могут быть также запущены от имени другого пользователя, отличного от того, под которым запускается агент сервера. Подробнее см. руководство администратора клиент-серверного варианта, [описание служебного файла swpuser.ini](#)

Область применения (уточнение): управляемое приложение, обычное приложение.

2. Запрещается выполнять запись каких-либо файлов в каталог исполняемых файлов 1С:Предприятия, получаемого с помощью метода **КаталогПрограммы**. Использование метода **КаталогПрограммы** допустимо только для чтения или запуска файлов. Например, при работе в ОС Windows, для запуска копии тонкого клиента 1С:Предприятия текущей версии, допустимо использовать:

ЗапуститьПриложение(КаталогПрограммы() + "1cv8s.exe");

локализация конфигурации на другие языки, следует обеспечивать переносимость файлов, сформированных из кода конфигурации, между различными операционными системами с различными кодировками. Для этого необходимо:

3.1. В именах файлов, автоматически формируемых из кода конфигурации, указывать только английские буквы, а также цифры, а в качестве кодировки текстовых файлов использовать только UTF-8 (именно этот формат предпочтителен, т.к. только с ним корректно работает операционная система macOS).

Это ограничение распространяется на файлы сообщений обмена, выгрузки данных, электронных документов и пр., которые автоматически формируются системой, в том числе на файлы, упакованные в архивы (например, zip). Исключение составляют случаи, когда на формат файлов невозможно повлиять, например, это формат сторонней системы.

3.2. В тех случаях, когда имя файла не генерируется системой, а его явно вводит пользователь, разрешить ввод русскоязычных имен, но при этом дать возможность транслитерировать его в англоязычное имя. По умолчанию, если это технически возможно и не снижает удобство работы, рекомендуется предлагать англоязычное имя файла, а для текстовых файлов – сохранение в кодировке UTF-8.

Также эти рекомендации по выбору имени и кодировки файла следует разместить в справке к тем местам программы, где пользователь имеет возможность сохранять файлы и выбирать кодировку.

В конфигурациях на базе **Библиотеки стандартных подсистем** для транслитерации имен файлов рекомендуется использовать функцию **СтроковыеФункцииКлиентСервер.СтрокаЛатиницей**.

Работа с временными файлами и каталогами

При необходимости использования временных файлов и каталогов необходимо соблюдать следующие требования:

1. Для получения имени временного файла следует использовать метод **ПолучитьИмяВременногоФайла** (исключение составляет веб-клиент, см. ниже п. 3). В противном случае возможна некорректная работа конфигурации в многопользовательском режиме, с включенными профилями безопасности, возникновение проблем с правами доступа к файлам операционной системы, а также неконтролируемое увеличение количества ненужных временных файлов, которые не будут своевременно удалены.

Например, неправильно:

ИмяПромежуточногофайла = "C:\Временные файлы 1С\TempFile.xml";
Данные.Записать(ИмяПромежуточногофайла);

У текущего пользователя может не быть прав на запись в указанный каталог. Кроме того, при одновременном выполнении этого кода из двух разных сеансов возникнет ошибка.

Правильно:

```
ИмяПромежуточногофайла = ПолучитьИмяВременногофайла("xml");
Данные.Записать(ИмяПромежуточногофайла);
```

При использовании этой функции будет получено уникальное имя, гарантирован доступ к файлу.

Кроме того, при использовании метода **ПолучитьИмяВременногоФайла** платформа **1С:Предприятие** сохраняет контроль над такими файлами и автоматически удаляет их при перезапуске рабочего процесса (если файл был создан на стороне сервера) или клиентского приложения (если файл был создан на стороне клиента).

Если же имя временного файла было сформировано каким-то другим способом, и прикладной код не удалил (либо по какой-то причине не смог удалить) ранее созданный временный файл, то платформа такой файл не контролирует, и он остается в файловой системе на неопределенное время. Накапливание «потерянных» временных файлов может представлять серьезную проблему, особенно для информационных баз с большим количеством активно работающих пользователей (например, при работе в режиме сервиса).

Таким образом, неправильно:

```
Каталог = КаталогВременныхФайлов();
ИмяФайла = Стока(Новый УникальныйИдентификатор) + ".xml";
ИмяПромежуточногофайла = Каталог + ИмяФайла;
Данные.Записать(ИмяПромежуточногофайла);
```

Если по каким-то причинам прикладной код не удалит созданный файл (например, между блоками создания и удаления временного файла возникнет штатное или нештатное исключение), этот файл так и останется в каталоге временных файлов.

Правильно:

```
ИмяПромежуточногофайла = ПолучитьИмяВременногофайла("xml");
Данные.Записать(ИмяПромежуточногофайла);
```

При использовании метода **ПолучитьИмяВременногоФайла** будет получено уникальное имя, гарантирован доступ к файлу, а также временный файл будет автоматически удален платформой **1С:Предприятие** после завершения рабочего процесса сервера или клиентского приложения.

2. Для создания временного каталога рекомендуется также использовать имя, полученное при помощи метода **ПолучитьИмяВременногоФайла** (исключение составляет веб-клиент, см. ниже п. 3). Это гарантирует уникальность имени создаваемого каталога при работе в многопользовательском режиме и гарантирует, что после перезапуска рабочего процесса или клиентского приложения временный каталог будет автоматически удален платформой **1С:Предприятие**. После этого, внутри созданного каталога можно создавать другие каталоги и файлы без ограничений.

3.1. При выполнении кода веб-клиентом метод **ПолучитьИмяВременногоФайла** недоступен. Поэтому для формирования имен временных файлов и каталогов необходимо использовать функцию **КаталогВременныхФайлов** и объект **УникальныйИдентификатор**.

Неправильно:

```
Каталог = КаталогВременныхФайлов();
ИмяФайла = "TempDataFile.xml";
ИмяПромежуточногофайла = Каталог + ИмяФайла;
Данные.Записать(ИмяПромежуточногофайла);
```

Правильно:

```
Каталог = КаталогВременныхФайлов();
ИмяФайла = Стока(Новый УникальныйИдентификатор) + ".xml";
ИмяПромежуточногофайла = Каталог + ИмяФайла;
Данные.Записать(ИмяПромежуточногофайла);
```

3.2. Если в конфигурацию встроена **Библиотека стандартных подсистем**, для создания временных каталогов на стороне клиента необходимо использовать процедуру **ФайловаяСистемаКлиент.СоздатьВременныйКаталог**.

4. После окончания работы с временным файлом или каталогом, его необходимо удалить самостоятельно. Нельзя рассчитывать на автоматическое удаление файлов и каталогов при следующем запуске платформы, это может привести к исчерпанию свободного места в каталоге временных файлов.

```
ИмяПромежуточногофайла = ПолучитьИмяВременногофайла("xml");
Данные.Записать(ИмяПромежуточногофайла);

// Работа с файлом
...

// Удаляем временный файл
Попытка
    УдалитьФайлы(ИмяПромежуточногофайла);
Исключение
    ЗаписьЖурналаРегистрации(НСтр("ru = 'Мой механизм.Действие'"), УровеньЖурналаРегистрации.Ошибка, , ,
ПодробноеПредставлениеОшибка(ИнформацияОб ошибке()));
КонецПопытки;
```

См. также: [Использование Журнала регистрации](#).

5. При использовании временных файлов и каталогов на сервере, необходимо полностью завершать работу с ними в рамках одного серверного вызова. При работе конфигурации с использованием кластера серверов, при следующем вызове эти файлы могут стать недоступны, так как код начнет исполняться на другом компьютере. При необходимости сохранить данные между серверными вызовами в пределах одного сеанса следует использовать временное хранилище платформы (методы **ПоместитьВоВременноеХранилище**, **ПолучитьИзВременногоХранилища**).

5.1. В редких случаях может возникнуть необходимость передачи данных во временных файлах между сеансами, например, при подготовке данных для фонового задания, при организации длительного процесса, обслуживающего несколько последовательных вызовов web-сервиса. Необходимо самостоятельно обеспечивать гарантировано общее место хранения, права для доступа к файлам из разных мест их обработки, удаление файлов по истечению сроков их обработки или аварийного завершения процесса обработки. Рекомендуется использовать следующий подход:

- Для обеспечения доступа со всех возможных мест обработки заводится константа для хранения общего пути к файлам, доступного для доступа со всех серверов кластера;
- При создании временных файлов их имена заносятся во вспомогательный регистр сведений с сохранением времени создания файла;
- При штатном прохождении процесса, последняя операция, которой были нужны файлы, перед своим завершением удаляет как сам файл, так и записи о них во вспомогательном регистре;
- Вспомогательное регламентное задание периодически проверяет наличие записей во вспомогательном регистре, время существования которых заранее превышает время штатного завершения процесса. При обнаружении таких записей, задание удаляет временные файлы и записи о них.

Передача файлов между клиентом и сервером

1. При одновременной работе с файлом на клиенте и на сервере необходимо использовать передачу файла через временное хранилище (методы **ПоместитьФайлы**, **ПолучитьФайлы**, **НачатьПомещениеФайла**, **ПоместитьВоВременноеХранилище**, **ПолучитьИзВременногоХранилища**). В общем случае клиент и серверы кластера - это разные компьютеры с разной файловой системой, причем доступ к файлам может происходить под разными пользователями ОС с различными правами.

Неправильно:

```

&НаКлиенте
Процедура ОбработатьФайл()
...
ИмяФайла = "C:\Файлы для обработки\Загрузка.xml";
Результат = ПроизвестиОбработкуНаСервере(ИмяФайла);
...

КонецПроцедуры

&НаСервере
Функция ПроизвестиОбработкуНаСервере(ИмяФайла)
    Чтение = Новый ЧтениеТекста(ИмяФайла);
    ...
    Результат = Чтение.Прочитать();
    Возврат Результат;

КонецФункции

Правильно:

&НаКлиенте
Процедура ОбработатьФайл()
...
ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";
ОписаниеОповещения = Новый ОписаниеОповещения(
    "ОбработатьФайлЗавершение", ЭтотОбъект);

НачатьПомещениеФайла(ОписаниеОповещения,
    ИмяФайлаДляОбработки, Ложь,
    УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте
Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)
...
Результат = ПроизвестиОбработкуНаСервере(Адрес);
...

КонецПроцедуры

&НаСервере
Функция ПроизвестиОбработкуНаСервере(Адрес)
Данные = ПолучитьИзВременногоХранилища(Адрес);
ИмяПромежуточногоФайла = ПолучитьИмяВременногоФайла("txt");
Данные.Записать(ИмяПромежуточногоФайла);

Чтение = Новый ЧтениеТекста(ИмяПромежуточногоФайла);
...
Результат = Чтение.Прочитать();
...

УдалитьФайлы(ИмяПромежуточногоФайла);

Возврат Результат;

КонецФункции

2. Для сохранения данных во временном хранилище между несколькими серверными вызовами, при помещении его в хранилище необходимо использовать параметр УникальныйИдентификаторФормы метода ПоместитьФайл, передав в него идентификатор текущей формы. Такие значения будут удалены из временного хранилища только при закрытии указанной формы. При этом, при повторном помещении того же файла во временное хранилище, предыдущее значение необходимо удалять вручную. Например:

Неправильно:

&НаКлиенте
Процедура ОбработатьФайл()
...
// Первый серверный вызов
ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";
ОписаниеОповещения = Новый ОписаниеОповещения(
    "ОбработатьФайлЗавершение", ЭтотОбъект);

НачатьПомещениеФайла(ОписаниеОповещения,
    ИмяФайлаДляОбработки, Ложь,
    УникальныйИдентификатор);

...
КонецПроцедуры

&НаКлиенте
Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)
...
Результат = ПроизвестиНачальнуюОбработкуНаСервере(Адрес);
ПродолжитьОбработкуФайла();
...

КонецПроцедуры

&НаКлиенте
Процедура ПродолжитьОбработкуФайла()
...
// Второй серверный вызов с той же версией файла
Результат = ПроизвестиПромежуточнуюОбработкуНаСервере(Адрес);
...

// Третий серверный вызов с новой версией файла
ОписаниеОповещения = Новый ОписаниеОповещения(
    "ПродолжитьОбработкуФайлаЗавершение", ЭтотОбъект);

```

```
НачатьПомещениефайла(ОписаниеОповещения,,  
    ИмяФайлаДляОбработки, Ложь,  
    УникальныйИдентификатор);
```

КонецПроцедуры

&НаКлиente

Процедура ПродолжитьОбработкуФайлаЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

```
...  
    Результат = ПроизвестиКонечнуюОбработкуНаСервере(Адрес);  
...
```

КонецПроцедуры

При этом во временном хранилище формы останется две копии файлов. Адрес второй копии будет находиться в переменной **Адрес**, а адрес первой копии будет утерян. Это приводит к затрате дополнительных ресурсов приложения, замедлению работы.

Правильно:

&НаКлиente

Процедура ОбработатьФайл()

```
...  
    // Первый серверный вызов  
    ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";  
  
    ОписаниеОповещения = Новый ОписаниеОповещения(  
        "ОбработатьФайлЗавершение", ЭтотОбъект);  
  
    НачатьПомещениефайла(ОписаниеОповещения,,  
        ИмяФайлаДляОбработки, Ложь,  
        УникальныйИдентификатор);  
...
```

КонецПроцедуры

&НаКлиente

Процедура ОбработатьФайлЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

```
...  
    Результат = ПроизвестиНачальнуюОбработкуНаСервере(Адрес);  
    ПродолжитьОбработкуФайла();  
...
```

КонецПроцедуры

&НаКлиente

Процедура ПродолжитьОбработкуФайла()

```
...  
    // Второй серверный вызов с той же версией файла  
    Результат = ПроизвестиПромежуточнуюОбработкуНаСервере(Адрес);  
...  
  
    // Третий серверный вызов с новой версией файла  
    УдалитьИзВременногоХранилища(Адрес);  
  
    ОписаниеОповещения = Новый ОписаниеОповещения(  
        "ПродолжитьОбработкуФайлаЗавершение", ЭтотОбъект);  
  
    НачатьПомещениефайла(ОписаниеОповещения,,  
        ИмяФайлаДляОбработки, Ложь,  
        УникальныйИдентификатор);
```

КонецПроцедуры

&НаКлиente

Процедура ПродолжитьОбработкуФайлаЗавершение(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)

```
...  
    Результат = ПроизвестиКонечнуюОбработкуНаСервере(Адрес);  
...
```

КонецПроцедуры

3. Если в конфигурацию встроена **Библиотека стандартных подсистем** для помещения файлов во временное хранилище необходимо использовать процедуры **ЗагрузитьФайл** и **ЗагрузитьФайлы** общего модуля **ФайловаяСистемаКлиент**. Для сохранения данных файла между несколькими серверными вызовами необходимо использовать свойство **ИдентификаторФормы** параметра **ПараметрыЗагрузки**:

&НаКлиente

Процедура ОбработатьФайл()

```
...  
    ИмяФайлаДляОбработки = "C:\Файлы для обработки\Загрузка.xml";  
    ОписаниеОповещения = Новый ОписаниеОповещения("ОбработатьФайлЗавершение", ЭтотОбъект);  
  
    ПараметрыЗагрузки = ФайловаяСистемаКлиент.ПараметрыЗагрузкиФайла();  
    ПараметрыЗагрузки.ИдентификаторФормы = УникальныйИдентификатор;  
    ПараметрыЗагрузки.Интерактивно = Ложь;
```

```
ФайловаяСистемаКлиент.ЗагрузитьФайл(ОписаниеОповещения,  
    ПараметрыЗагрузки, ИмяФайлаДляОбработки);
```

КонецПроцедуры

&НаКлиente

Процедура ОбработатьФайлЗавершение(ПомещенныйФайл, ДополнительныеПараметры)

```
...  
    Результат = ПроизвестиОбработкуНаСервере(Адрес);  
...
```

КонецПроцедуры

См. также

- [Установка внешних компонент и расширений платформы](#)
- [Общие требования к конфигурации](#)

Оптимизация использования оперативной памяти

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

#std725

Методическая рекомендация (полезный совет)

1. Не следует разрабатывать решения исходя из неограниченного объема оперативной памяти. Для многопользовательских систем любое неэффективное использование памяти может катастрофически сказаться на работоспособности.

Следует избегать формирования больших структур данных в памяти. Если объем данных, с которыми работает бизнес-логика, сам по себе ничем не ограничен, его нужно ограничивать искусственно, обрабатывая данные порциями и сохраняя результаты в базу или файлы.

2. При потенциально неограниченных выборках данных из ИБ следует получать данные из базы порциями фиксированного размера.
Например, неправильно:

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
| Номенклатура.Ссылка,
| Номенклатура.Наименование,
| Номенклатура.ВидНоменклатуры
|Из
| Справочник.Номенклатура КАК Номенклатура";

// Выгрузка всего справочника в таблицу значений
Номенклатура = Запрос.Выполнить().Выгрузить();
Для каждого ПозицияНоменклатуры Из Номенклатура Цикл
// Обработка элемента справочника
// ...
КонецЦикла;
```

поскольку весь результат запроса сразу помещается в память, в таблицу значений.

Также неправильно:

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
| Номенклатура.Ссылка,
| Номенклатура.Наименование,
| Номенклатура.ВидНоменклатуры
|Из
| Справочник.Номенклатура КАК Номенклатура";

РезультатЗапроса = Запрос.Выполнить();
// Обход результата запроса
ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
// Обработка элемента выборки
// ...
КонецЦикла;
```

поскольку и в этом случае при выполнении запроса его результат будет сначала считан в память целиком (*).

* Примечание. Если используется 32-битная версия платформы, и размер результата запроса превосходит размер имеющейся памяти, то данные будут записаны на диск, а затем считаны оттуда в процессе вызовов Выборка.Следующий().

Правильно ограничивать результат запроса искусственно:

```
ВсеОбработано = Ложь;
Пока Истина Цикл
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ ПЕРВЫЕ 1000
| Номенклатура.Ссылка,
| Номенклатура.Наименование,
| Номенклатура.ВидНоменклатуры
|Из
| Справочник.Номенклатура КАК Номенклатура
|ГДЕ
| <условие выборки необработанных записей>";

РезультатЗапроса = Запрос.Выполнить();
ВсеОбработано = РезультатЗапроса.Пустой();
Если ВсеОбработано Тогда
    Прервать;
КонецЕсли;

// Обход порции результата запроса
ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
// Обработка элемента выборки
// ...
КонецЦикла;

КонецЦикла;
```

Также правильно:

```
Выборка = Справочники.Номенклатура.Выбрать(..., Отбор);
Пока Выборка.Следующий() Цикл
// Обработка элемента выборки
// ...
КонецЦикла;
```

поскольку в этом случае платформа 1С:Предприятие выполняет получение данных из базы порциями фиксированного размера.

Кроме того, число элементов выборки автоматически ограничивает платформа 1С:Предприятие в запросах динамических списков.

3. Недопустимо работать с большими XML документами с помощью объектов встроенного языка, предназначенных для обработки файлов целиком: текстовые документы в **ТекстовыйДокумент**, XML в **ДокументDOM** и HTML в **ДокументHTML**, а также создавать в памяти XDTO-пакеты размером с весь XML-файл целиком.

В противном случае, весь файл загружается в оперативную память целиком. Исключения составляют отдельные случаи, когда необходим произвольный доступ к содержимому файла, к какой-то конкретной его части.

Следует использовать объекты для последовательной записи и последовательного чтения: **ЧтениеXML**, **ЧтениеТекста**, **ЗаписьXML**, **ЗаписьТекста**, с помощью которых можно прочитать файл порциями и расходовать память экономно.

При использовании механизмов XDTO неправильно зачитывать в память весь XML-файл целиком (**ФабрикаXDTO.ПрочитатьXML(ЧтениеXML)**). Вместо этого следует зачитывать XML-файл последовательно, с помощью объекта **ЧтениеXML**, а его отдельные фрагменты (теги) десериализовывать с помощью фабрики XDTO.

4. Другая распространенная причина неэффективное использование памяти - утечки памяти. К утечкам памяти приводят создание циклических ссылок – память выделяется и не освобождается. Например, если есть объекты, внутри которых вложены другие объекты, и где-то в глубине они ссылаются на самый верхний объект. В результате образуется циклическая ссылка.

Упрощенный пример циклической ссылки:

```
Данные = Новый Структура;
Данные.Вставить("Ключ", Данные);
```

Следует разрывать (очищать) ссылки, когда объект становится не нужен.

Например, для примера выше:

```
Данные.Ключ = Неопределено;
```

Для выявления утечек памяти можно применять технологический журнал, включив в файл настройки параметров технологического журнала logcfg.xml элемент <Leaks>.

Подробнее см.:

- [Документация к платформе 1С:Предприятие. Приложение 3. Описание и расположение служебных файлов - logcfg.xml](#)
- [Поиск циклических ссылок](#)

5. Чрезмерное (неоправданное) применение [общих модулей с повторным использованием возвращаемых значений](#) может также приводить к излишнему потреблению памяти.

Таймауты при работе с внешними ресурсами

Область применения: управляемое приложение, обычное приложение.

#std748

1. При работе с внешними ресурсами с помощью объектов **WSОпределения**, **WSПрокси**, **HTTРСоединение**, **FTPСоединение**, **ИнтернетПочтовыйПрофиль** следует задавать таймаут – предельное время ожидания выполнения операции. В противном случае, в результате бесконечного ожидания программа зависнет или часть функционала программы станет недоступна.

Установка таймаута является защитой от целого ряда внешних факторов:

- нестабильного подключения к Интернету, когда регулярно происходит прерывание связи, и система не может получить целевой ответ сервера, к которому выполняется подключение;
- при включенных антивирусных программах или при неправильных настройках брандмауэра;
- неправильной настройки прокси-сервера;
- ненадежной работы веб-сервера из-за возросшей нагрузки или некорректной работы скриптов.

Например, при получении описания веб-сервиса и вызове его операций – если удаленная сторона долго не отвечает (например, выключена, находится на обслуживании или возникли временные неполадки), ожидание ответа может длиться бесконечно. Поэтому если веб-сервис был вызван в результате интерактивных действий пользователя, то внешне будет выглядеть так, что «программа зависла»; а если веб-сервис вызывается из регламентного задания, то связанная с ним часть функционала программы может стать недоступна.

2. В общем виде, время выполнения операции с внешними ресурсами складывается из шести этапов:

- DNS Lookup — время, потраченное на определение IP адреса по доменному имени (если применимо);
- Connect — установка соединения с веб-сервером по полученному IP-адресу;
- Send — отправка данных на веб-сервер;
- Wait — ждем, пока данные дойдут до веб-сервера и он их обработает;
- Receive — получение ответа от веб-сервера;
- Cache Read — получение данных от веб-сервера.

Например, при таймауте в 60 секунд программа и вызываемый внешний ресурс должны успеть выполнить шесть выше перечисленных этапов операции, иначе соединение будет разорвано, а передача данных прервана. Однако если в процессе выполнения операции возникнет сбой, то система и/или пользователь будет зря ожидать 60 секунд.

Поэтому величину таймаута рекомендуется определять, исходя из ожидаемого времени выполнения конкретной операции:

- Для быстрых операций (например, проверка доступности сервера) величина таймаута должна выбираться, соответственно, небольшой;
- В общем случае, не следует выбирать таймаут более 3 минут, чтобы при недоступности удаленной стороны не допустить эффект «зависания» программы;
- Но если операция выполняется долго из-за этапов Send или Cache Read, т.е. это передача больших объемов данных на веб-сервер или загрузка большого файла с внешнего ресурса, то следует устанавливать большой таймаут, исходя из оценки объема передаваемых данных, но не более 12 часов.

Подробнее о рекомендуемых величинах таймаута для различных операций см. в таблице п. 4.

3. Рекомендации по снижению величин таймаута и повышению отзывчивости программы при работе с внешними ресурсами.

3.1. При разработке веб-сервисов, на операции которых предусмотрен таймаут более 20 секунд (ориентировочно), рекомендуется:

- предусмотреть в веб-сервисе отдельную контрольную операцию Ping;
- при работе с этим веб-сервисом, предварительно получать для нее прокси с небольшим таймаутом в 7 секунд и вызывать контрольную операцию Ping;
- только после этого получать основной прокси.

Пример вызова веб-сервиса.

Неправильно

Реализация модуля веб-сервиса PingPong:

```
Функция Pong(Знач Параметр)
    Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru
= 'Привет, %1'"), Параметр);
КонецФункции
```

Правильно

```
Функция Ping()
    Возврат Истина; // Проверка связи
КонецФункции
```

```
Функция Pong(Знач Параметр)
```

```
    Возврат
СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ru
= 'Привет, %1'"), Параметр);
КонецФункции
```

Реализация вызывающей стороны (без использования Библиотеки стандартных подсистем):

```
// Ждем не более 3 минуты
PingPong = Новый WSПрокси(АдресВебСервиса, , , , , 60);
Результат = PingPong.Pong(НСтр("ру = 'Мяч'"));
// Проверка связи
PingPong.Ping();
```

```
// Сервис жив, далее работаем с ним и ждем не более минуты
PingPong = Новый WSПрокси(АдресВебСервиса, ' ', ' ', 60);
Результат = PingPong.Pong(НСтр("ru = 'Мяч'"));
```

При использовании Библиотеки стандартных подсистем:

- для работы с веб-сервисами предназначена функция WSПрокси общего модуля ОбщегоНазначения (включает в себя поддержку контрольной операции **Ping**);
- для получения данных по протоколам HTTP(S) и FTP(S) – подсистема «Получение файлов через Интернет».

Пример реализации вызывающей стороны с использованием Библиотеки стандартных подсистем:

```
// Сделать контрольный вызов Ping и ждать не более минуты на дальнейших операциях.
PingPong = ОбщегоНазначения.WSПрокси(АдресВебСервиса, ..., 60, Истина);
// Сервис точно жив, далее работаем с ним.
Результат = PingPong.Pong(НСтр("ru = 'Мяч'"));
```

3.2. Для других видов внешних ресурсов (не веб-сервисов) рекомендуется применять аналоги операции **Ping**. Например:

- для сервисов, работающих через REST API – это контрольная отправка тестовой команды; в большинстве случаев, если ответ с кодом 200, то сервис работает;
- для FTP/WebDAV-ресурсов – это контрольная загрузка (отправка) файла-пустышки.

3.3. Веб-сервисы, операции, которых занимают объективно много времени из-за этапа Wait (т.е. долго отрабатывает само веб-приложение), и они не могут быть ускорены (оптимизированы) по объективным причинам, следует переводить на асинхронный режим выполнения:

- запускать фоновое задание для выполнения подобной «тяжелой» операции;
- и предусмотреть дополнительные операции по проверке готовности и получению результата.

Пример асинхронного вызова веб-сервиса.

Неправильно

Реализация модуля веб-сервиса Long:

```
Функция GetData()
    Результат = <очень длительные вычисления>;
    Возврат Результат;
Конецфункции
```

Правильно

```
Функция StartDoLong()
    // запуск фонового задания
    ИдентификаторОперации = ...
    // возвращаем идентификатор операции для отслеживания ее готовности
    Возврат ИдентификаторОперации;
Конецфункции
```

```
Функция IsReady(Знач ИдентификаторОперации)
    // проверяем, завершено ли фоновое задание по переданному
    идентификатору
    Готовность = ...
    Возврат Готовность;
Конецфункции
```

```
Функция GetData(Знач ИдентификаторОперации)
    Результат = <получаем уже готовый результат по переданному
    идентификатору>;
    Возврат Результат;
Конецфункции
```

```
Long = Новый WSПрокси(АдресВебСервиса, ..., 600); // ждем 1 час
ИдентификаторОперации = Long.StartDoLong();
```

```
Пока Не Long.IsReady(ИдентификаторОперации) Цикл
    <ждем определенный интервал времени>
КонецЦикла;
Результат = Long.GetData(ИдентификаторОперации);
```

* это лишь упрощенная схема реализации вызывающей стороны; в действительности, код вызывающей стороны также должен быть реализован асинхронно с помощью включения регламентного задания, либо периодического обработчика ожидания на клиенте, который проверяет готовность и получает результат.

4. Рекомендуемые величины таймаутов для различных операций:

Операция	Таймаут (секунд)
Получение описания веб-сервиса	7
Проверка корректности введенного адреса, взаимодействие с менеджером сервиса в модели сервиса и прочие «быстрые» операции	10-20
Получение сведений об одном контрагенте, обмен сообщениями, отправка SMS, удаленное администрирование ИБ в модели сервиса	60-120 ¹
Передача сообщений обмена данными через веб-сервис или получение файлов из внешнего ресурса до 1 Мб.	120-180 ¹
Загрузка файлов более 1 Мб	Если известен размер файла, то размер в мегабайтах * 128 ² , иначе предельное время загрузки, но не более 43200 ³

¹ Следует вызывать только после контрольной операции **Ping**.

² Загрузка 1 мегабайта данных занимает 128 секунд, при скорости 64 кбит/с, т.к. сотовые операторы в определенных случаях ограничивают скорость загрузки этой величиной.

³ Таймаут продолжительностью 43200(12 часов) сек. является компромиссным решением, т.к. в случае нештатной ситуации процесс «отвиснет» на следующее утро и вернет управление, в отличие от полностью зависнувшей программы при неустановленном таймауте.

Безопасность прикладного программного интерфейса сервера

При работе в режиме управляемого приложения, клиентское приложение (тонкий или веб-клиент) обращается к серверу 1С:Предприятия посредством открытого HTTP-протокола. Таким образом, сервер 1С:Предприятия может быть вызван извне сторонними программами тем же способом, как это штатно делает клиентское приложение, и злоумышленник может получить несанкционированный доступ к пользовательским данным, нарушить работоспособность сервера.

1. Несанкционированный вызов серверного кода конфигурации с клиента.

1.1. Потенциальную угрозу безопасности представляют все серверные процедуры и функции, доступные для вызова из клиентского кода. Они составляют прикладной программный интерфейс сервера 1С:Предприятия. К ним, как правило, относятся:

- Экспортные процедуры и функции, размещенные в общих модулях с признаком "Сервер" и "Вызов сервера". Вызов таких процедур и функций возможен напрямую с клиента.

Подробнее см. [Ограничение на установку признака "Вызов сервера" у общих модулей](#)

- Все процедуры и функции модулей форм объектов с директивами компиляции &НаСервере, &НаСервереБезКонтекста. Вызов таких процедур и функций доступен из контекста клиента после успешного получения формы, даже если эти процедуры и функции не экспортные. Это делает возможным вызов кода в контексте, который не предполагался разработчиком.

Например, код модуля формы **Справочник.Сотрудники.ФормаЕлемента**:

&НаКлиенте
Процедура УволитьСотрудника(Команда)

```
Если ДатаРегистрацииУвольнения > ДатаЗапрета Тогда
    ЗарегистрироватьУвольнение();
КонецЕсли;
```

КонецПроцедуры

&НаСервере
Процедура ЗарегистрироватьУвольнение()
...
КонецПроцедуры

Пример стороннего кода, вызывающий напрямую серверную процедуру для обхода проверки, предусмотренной разработчиком формы в обработке команды **УволитьСотрудника**:

```
ПараметрыФормы = Новый Структура("Ключ", ВыбранныйСотрудник)
Форма = ПолучитьФорму("Справочник.Сотрудники.ФормаЕлемента", ПараметрыФормы);
Форма.ЗарегистрироватьУвольнение();
```

1.2. В общем случае не рекомендуется размещать в серверных процедурах и функциях модулей форм код, обеспечивающий бизнес-логику, и который не относится к клиент-серверному взаимодействию и обработке реквизитов формы.

1.3. Особого внимания требуют серверные процедуры и функции, использующие [установку привилегированного режима](#), или размещенные в общих модулях с признаком **Привилегированный**.

2. Проникновение небезопасного кода на сервер и его выполнение.

Любые возможности конфигурации по выполнению "внешнего" кода или произвольных текстов запросов на сервере, не являющихся частью самого прикладного решения, представляют серьезную опасность.

Также опасны внешние отчеты и обработки, СОМ-объекты и внешние компоненты. В частности, код внешних обработок может непосредственно обращаться ко всем общим модулям без признака "Вызов сервера", к модулям объектов и менеджеров объектов конфигурации, пытаясь переходить в привилегированный режим.

Такие возможности создают прямую угрозу работоспособности сервера из-за некорректного или преднамеренно вредоносного кода: порчу или похищение данных, зависание или остановка рабочего процесса из-за зацикливания, утечек памяти, ресурсоемких операций и запросов и т.д.

Подробнее см. [Ограничение на выполнение "внешнего" кода](#).

3. Клиентское приложение (тонкий клиент или веб-браузер) не гарантирует безопасность данных, переданных на сторону клиента. Эти данные легко могут быть перехвачены и прочитаны вредоносным программным обеспечением, установленным на клиентском компьютере.

Серверные процедуры и функции должны возвращать в форму только окончательный результат расчета. Следует избегать передачи в форму исходных или промежуточных данных, которые могут раскрывать побочную, возможно приватную информацию бизнес-процесса.

Ограничение на установку признака «Вызов сервера» у общих модулей

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

#std679

1. Не следует всем общим модулям с признаком **Сервер** принудительно устанавливать флагок **Вызов сервера**. В таких общих модулях следует размещать только те процедуры и функции, которые действительно предназначены для вызова из клиентского кода и гарантируют выполнение только тех действий (и передачи только тех данных на сторону клиента), которые разрешены пользователю при его работе в программе. Например, серверная функция, реализующая некоторый алгоритм расчета, должна передавать на сторону клиента окончательный результат этого расчета, но не исходные (или промежуточные) данные для расчета, которые сами по себе могут быть недоступны текущему пользователю.

См. также: [Безопасность прикладного программного интерфейса сервера](#), [Правила создания общих модулей](#)

Особого внимания требуют процедуры и функции, использующие установку [привилегированного режима](#), или размещенные в общих модулях с признаком **Привилегированный**.

Серверные процедуры и функции, не предназначенные для вызова из клиентского кода, следует размещать в общих модулях без признака **Вызов сервера**.

2.1. Как правило, при разработке объектов конфигурации (справочников, документов и пр.) исходят из того, что в управляемом режиме работа с экземплярами этих объектов (**СправочникОбъект**, **ДокументОбъект** и т.д.) выполняется на стороне сервера. Поэтому в управляемом режиме не гарантируется возможность работы с ними на стороне клиентского приложения.

В частности, в толстом клиенте в режиме управляемого приложения не следует создавать или получать объекты:

- в клиентских общих модулях (пользуясь инструкцией препроцессора **ТолстыйКлиентУправляемоеПриложение**);
- в обычных формах при запуске в управляемом режиме. Такие формы следует использовать только в режиме обычного приложения.

Это позволит избежать выполнения кода модулей объектов и подписок на события на клиенте, а также избыточных серверных вызовов процедур и функций общих модулей из этого кода.

См. также: [Поддержка толстого клиента, управляемое приложение, клиент-сервер](#)

2.2. Если конфигурация не рассчитана на работу в толстом клиенте, управляемое приложение, следует снять флагок **Толстый клиент (управляемое приложение, режим клиент-сервер)**, для того чтобы при проверке конфигурации избежать ложных сообщений об ошибках.

Безопасное хранение паролей

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

#std740

1. При разработке подсистем, взаимодействующих с различными внешними ресурсами (электронной почтой, веб-сервисами, FTP-ресурсами и т.п.) возникает необходимость запрашивать и передавать данные аутентификации к этим ресурсам: логин и пароль.

2. Для сведения к минимуму возможности перехвата пароля злоумышленниками не следует хранить пароли и другую конфиденциальную информацию в информационной базе. При этом минимальный уровень защищенности – в файловых информационных базах, в которых файл базы может быть скопирован целиком любым пользователем информационной базы. В клиент-серверной информационной базе доступ к базе данных, как правило, имеется только у администраторов СУБД.

Таким образом, следует запрашивать логин и пароль у пользователя и передавать их сразу, не сохраняя в информационной базе.

3. В ряде случаев такая схема работы доставляет объективные неудобства или принципиально невозможна:

- интерактивный запрос логина и пароля на каждую операцию может создавать значительный дискомфорт от работы, а временного сохранения на стороне клиента недостаточно;
- взаимодействие с различными внешними ресурсами должно выполняться на сервере, не зависимо от интенсивности работы пользователей с программой.

В таких случаях допустимо организовать хранение паролей и другой конфиденциальной информации в информационной базе, предупредив пользователей о последствиях. Следует помнить, что подобное хранение паролей не решает всех проблем безопасности, а лишь усложняет задачу для злоумышленника.

3.1. При этом не следует хранить пароли и другую конфиденциальную информацию в реквизитах тех же объектов метаданных, с которыми ведется повседневная работа. Для хранения такой информации следует использовать отдельный объект метаданных (например, регистр сведений), организовав к нему безопасный доступ на уровне системы прав доступа 1С:Предприятия.

3.2. При использовании **Библиотеки стандартных подсистем (БСП)** следует использовать безопасное хранилище паролей, которое решает ряд задач:

- Имея доступ к объекту метаданных, пользователь может прочитать содержимое реквизита с паролем, что невозможно при использовании безопасного хранилища. Для исключения случаев несанкционированного доступа к безопасному хранилищу получение и запись данных (паролей) возможна только в привилегированном режиме.
- Данные в безопасном хранилище хранятся в закрытом виде и тем самым исключаются случаи непредумышленной «засветки» паролей.
- Безопасное хранилище исключено из планов обмена, что предотвращает утечку паролей из информационной базы при обмене данными.

Для работы с безопасным хранилищем паролей предназначены процедуры и функции общего модуля **ОбщегоНазначения**:

ЗаписатьДанныеВБезопасноеХранилище, **ПрочитатьДанныеИзБезопасногоХранилища** и **УдалитьДанныеИзБезопасногоХранилища**. Подробнее см. комментарии к этим функциям в БСП и раздел «3.4. Базовая функциональность - Использование при разработке конфигурации - Безопасное хранилище паролей» документации БСП.

3.3. Не следует хранить пароли в реквизитах формы, их следует извлекать только на стороне сервера и непосредственно перед их использованием. В противном случае, при открытии формы с маскированным вводом (или просмотром) пароля, пароль передается с сервера на клиент в открытом виде, что делает возможным его перехват. Установка привилегированного режима производится непосредственно перед вызовом функций, а не внутри них, что бы исключить получение или запись любых паролей в сеансе с любыми правами. Безопасность вызова должен обеспечивать вызывающий код, который обращается к конкретным паролям.

Для маскировки пароля на форме в обработчике событий формы **ПриСозданииНаСервере** необходимо разместить следующий код:

```
УстановитьПривилегированныйРежим(Истина);
Пароли = ОбщегоНазначения.ПрочитатьДанныеИзБезопасногоХранилища(Объект.Ссылка, "Пароль", ПарольSMTP); // Пароль, ПарольSMTP – ключи соответствия данных в безопасном хранилище
УстановитьПривилегированныйРежим(Ложь);
```

```
Пароль = ?(ЗначениеЗаполнено(Пароли.Пароль), ЭтотОбъект.УникальныйИдентификатор, "");
ПарольSMTP = ?(ЗначениеЗаполнено(Пароли.ПарольSMTP), ЭтотОбъект.УникальныйИдентификатор, "");
```

В обработчике события формы **ПриЗаписиНаСервере**:

```
Если ПарольИзменен Тогда
    УстановитьПривилегированныйРежим(Истина);
    ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, Пароль);
    УстановитьПривилегированныйРежим(Ложь);
    Пароль = ?(ЗначениеЗаполнено(Пароль), ЭтотОбъект.УникальныйИдентификатор, "");
КонецЕсли;
```

```
Если ПарольSMTPИзменен Тогда
    УстановитьПривилегированныйРежим(Истина);
    ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, ПарольSMTP, "ПарольSMTP");
    УстановитьПривилегированныйРежим(Ложь);
    ПарольSMTP = ?(ЗначениеЗаполнено(ПарольSMTP), ЭтотОбъект.УникальныйИдентификатор, "");
КонецЕсли;
```

где **Пароль** и **ПарольSMTP** - реквизиты формы. Если пароль ранее был сохранен в программе, то следует присвоить соответствующему реквизиту уникальный идентификатор формы, эмулирующий наличие пароля. При записи объекта в форме, если был введен новый пароль, то записываем его в объект, а реквизит формы вновь затираем уникальным идентификатором.

Ограничение на выполнение «внешнего» кода

Область применения: управляемое приложение, обычное приложение.

#std669

Помимо программного кода конфигурации, в прикладном решении может исполняться сторонний программный код, который может быть подключен с помощью внешних отчетов, внешних обработок, расширенных конфигураций, внешних компонент или другими способами (далее – внешний код). При этом злоумышленник может предусмотреть в нем различные деструктивные действия (как в самом внешнем коде, так и опосредованно, через запуск внешних приложений, внешних компонент, COM-объектов), которые могут нанести вред компьютерам пользователей, серверным компьютерам, а также данным в программе. Пример такой уязвимости: <https://1c.ru/news/info.jsp?id=21537>

Перечисленные проблемы безопасности особенно критичны при работе конфигураций в **модели сервиса**. Например, получив доступ к сервису, вредоносный код может получить доступ сразу ко всем приложениям всех пользователей сервиса.

1. Для прикладных решений запрещено выполнение в небезопасном режиме любого кода на сервере 1С:Предприятия, который не является частью самого прикладного решения (конфигурации). Ограничение не распространяется на код, прошедший аудит, и на код, выполняемый на клиенте.

Примеры недопустимого выполнения «внешнего» кода в небезопасном режиме:

- внешние отчеты и обработки (печатные формы и т.п.), расширения конфигурации, внешние компоненты и любые другие аналогичные возможности, с помощью которых пользователи подключают к конфигурации внешний код;
- алгоритмы на встроенным языке, тексты запросов или их фрагменты, которые пользователи интерактивно вводят в режиме 1С:Предприятия, и которые затем передаются в методы глобального контекста **Выполнить** или **Вычислить** (см. [«Ограничения на использование Выполнить и Вычислить на сервере»](#));
- изменение пользователями схем компоновки данных в отчетах, в которых разрешено использование внешних функций (эта возможность закрыта при использовании стандартной формы отчета: она не позволяет пользователям изменять схему компоновки данных, а из пользовательских полей использовать функции общих модулей нельзя). В том числе, возможность загрузки пользователями схем компоновки данных из внешних файлов.

При использовании в конфигурации Библиотеки стандартных подсистем (БСП) внешний код допустимо подключать только через соответствующие подсистемы БСП:

- как расширения конфигурации – с помощью средств подсистемы «Базовая функциональность»;
- как внешние отчеты и обработки – через «Дополнительные отчеты и обработки»;
- в виде внешних компонент – через подсистему «Внешние компоненты»;
- для запуска внешних программ – см. [«Безопасность запуска приложений»](#).

При этом указанное в этом пункте требование будет выполнено.

2. По умолчанию, в конфигурации для всех категорий пользователей должна быть отключена возможность интерактивно открывать внешние отчеты и обработки через меню Файл – Открыть. См. пп. 2.2 и 2.3 [«Стандартные роли»](#).

При этом в настройках программы должна быть предусмотрена обратная возможность разрешить это действие. В случае если администратор разрешает интерактивно открывать внешние отчеты и обработки, то информировать его и пользователей о том, что при открытии файлов внешних отчетов и обработок следует обращать особое внимание на их источник и не открывать файлы, полученные из источников, с которыми нет договоренности о разработке таких отчетов и обработок.

При использовании в конфигурации **Библиотеки стандартных подсистем** отключение интерактивного открытия внешних отчетов и обработок, настройка, а также соответствующие предупреждения уже предусмотрены.

3. Предупреждать администраторов об опасности перед подключением любого внешнего кода.

3.1. Выводимая информация должна включать в себя в явном виде сведения, что внешний код, полученный из недостоверных источников (с которыми, например, нет договоренности о разработке такого кода), может нанести вред компьютерам пользователей, серверным компьютерам, а также данным в программе. При этом администратор должен иметь возможность отказаться от загрузки внешнего кода (а также возможно повторить его загрузку позднее после проведения соответствующего аудита).

При использовании в конфигурации Библиотеки стандартных подсистем такие предупреждения для администратора уже предусмотрены в соответствующих подсистемах.

3.2. В то же время, остальные пользователи программы не должны получать дополнительных предупреждений при исполнении внешнего кода, подключение которого ранее было явно подтверждено администратором.

Для программного отключения см. раздел 7.10.2. [Отключение механизма защиты от опасных действий](#) в документации к платформе 1С:Предприятие.

При использовании в конфигурации **Библиотеки стандартных подсистем**

- подобное отключение предупреждений уже предусмотрено в соответствующих подсистемах;
- запрещено отключать предупреждения об опасных действиях во всех остальных случаях.

4. Если в конфигурации предусмотрены средства обновления конфигурации (из файлов .cf, .cfl), восстановления из резервной копии или загрузки из dt-файла в режиме 1С:Предприятия, то эти операции должны выполняться с соблюдением следующих правил:

- обновление должно быть доступно только пользователю с ролью «Администратор системы»;
- такое обновление должно выполняться только интерактивно текущим пользователем, а не служебным пользователем с полными правами;
- перед обновлением конфигурации из файла или восстановления из резервной копии, администратору должно показываться предупреждение о том, что он должен убедиться, что файл обновления получен из надежного источника;
- при обновлении конфигурации через Интернет, должно использоваться защищенное соединение (см. п. 7) и надежный источник, о чем нужно предупредить пользователя, когда он настраивает параметры подключения к источнику обновления.

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) операции обновления конфигурации и восстановления из резервной копии следует выполнять только средствами подсистем «Обновление конфигурации» и «Резервное копирование ИБ» БСП. При этом автоматически будут выполнены все требования, перечисленные выше в этом пункте.

5. Если в конфигурации предусмотрены средства загрузки произвольных файлов в программу, то следует также иметь в виду, что они могут содержать вредоносный исполняемый код. В этом случае в конфигурации следует предусмотреть

- для администратора – дополнительные средства контроля, в частности, список разрешенных (запрещенных) расширенных файлов для загрузки в программу;
- блокирование открытия исполняемых файлов из программы (даже если их разрешено загружать и хранить в программе).

Примечание: в общем случае, вредоносный код может содержаться даже в неисполняемых файлах, например, макровирусы в документах Microsoft Office. Однако в этом случае необходимые предупреждение об опасных действиях уже предусмотрены в сторонних приложениях Microsoft Office, поэтому в конфигурации не требуется предпринимать дополнительных мер защиты. Исключение составляет случай открытия через СОМ – см. [Безопасность программного обеспечения, вызываемого через открытые интерфейсы](#).

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) работу с файлами следует организовывать только средствами подсистемы «Работа с файлами». При этом автоматически будут выполнены все требования, перечисленные выше в этом пункте.

6. Безопасность внешних компонент.

6.1. Внешние компоненты, не являющиеся частью конфигурации (не размещенные в макетах конфигурации) потенциально опасны и их не следует загружать из источников, к которым нет доверия, с целью последующей установки и подключения. Пользователи без административных прав не должны иметь возможности загрузки, установки и подключения внешних компонент на сервере прикладного решения. При этом пользователю всегда должен задаваться вопрос и предоставляться выбор, устанавливать ли внешний компонент на клиенте.

Невыполнение этих требований может нарушить работоспособность и безопасность прикладного решения, серверов на которых оно работает и компьютера пользователя.

6.2. Сторонние внешние компоненты следует хранить в специальном справочнике, доступ на запись к которому есть только у администратора и подключать их только по навигационной ссылке на реквизит справочника, в котором хранятся двоичные данные компоненты.

Не следует подключать сторонние внешние компоненты по имени файла или по идентификатору программы, т.к. в этом случае злоумышленник сможет подменить путь к файлу или идентификатор программы и подключить свою вредоносную компоненту.

6.3. Внешние компоненты, входящие в состав конфигурации, должны храниться в макетах типа «Внешняя компонента». Данный тип макета [не локализуется](#).

6.4. При использовании в конфигурации **Библиотеки стандартных подсистем**, следует использовать методы подключения компонент библиотеки и полностью исключить непосредственное использование платформенных механизмов подключения внешних компонент, таких как:

- ПодключитьВнешнююКомпоненту;
- НачатьУстановкуВнешнейКомпоненты;
- УстановитьВнешнююКомпоненту;
- НачатьПодключениеВнешнейКомпоненты;
- ЗагрузитьВнешнююКомпоненту.

Для подключения компоненты из макета в составе конфигурации на клиенте следует использовать:

ОбщегоНазначенияКлиент . ПодключитьКомпонентуИзМакета

Для подключения компонент из хранилища внешних компонент (специального справочника с возможностью обновлять компоненты независимо от обновления конфигурации), следует использовать подсистему Внешние компоненты в Библиотеке стандартных подсистем:

ВнешниеКомпонентыКлиент . ПодключитьКомпоненту

7. При загрузке внешнего кода из удаленных источников в конфигурацию, следует:

- использовать только надежные источники, к которым есть доверие;
- выполнять передачу данных только по защищенным каналам связи.

ЗащищенноеСоединение = Новый ЗащищенноеСоединениеOpenSSL();
Соединение = Новый HTTPСоединение(Сервер,,,,,, ЗащищенноеСоединение);

При использовании в конфигурации **Библиотеки стандартных подсистем** необходимо использовать функцию **НовоеЗащищенноеСоединение** общего модуля **ОбщегоНазначенияКлиентСервер**:

ЗащищенноеСоединение = ОбщегоНазначенияКлиентСервер.НовоеЗащищенноеСоединение();
Соединение = Новый HTTPСоединение(Сервер,,,,,, ЗащищенноеСоединение);

См. также

- [Ограничения на использование Выполнить и Вычислить на сервере](#)
- [Облачные технологии](#) (статья на сайте 1c.ru)
- [Безопасность прикладного программного интерфейса сервера](#)

Ограничения на использование Выполнить и Вычислить на сервере

Область применения: управляемое приложение, обычное приложение.

#std770

1. При разработке решений следует учитывать, что опасно не только непосредственное выполнение кода, написанного в режиме **Предприятие**, но и те места, где методами **Выполнить** или **Вычислить** исполняется код, сконструированный на основе параметров, переданных в серверные функции и процедуры. Ограничение не распространяется на код, выполняемый на клиенте.

Например, код написан следующим образом:

- в клиентской функции в форме создается структура, в которую вставляется строка, написанная разработчиком конфигурации;
- клиентская функция передает эту структуру в серверную функцию формы;
- серверная функция формы вызывает серверную функцию общего модуля;
- в серверной функции исполняется код из строки, вставленной в структуру.

В этом случае не выполняется код, который интерактивно вводят пользователь, но, тем не менее, есть следующая уязвимость:

- злоумышленник создает структуру, в которую вставляет строку с вредоносным кодом;
- злоумышленник вызывает клиентскую функцию формы и таким образом исполняет вредоносный код на сервере.

Еще опаснее, если методы, в которых с помощью **Выполнить** или **Вычислить** исполняется код, принимаемый из параметров, будут располагаться в модулях с установленным признаком **ВызовСервера**.

2. Для исключения описанных уязвимостей, нужно в серверных процедурах и функциях вызов методов **Выполнить** или **Вычислить** предварять включением безопасного режима:

УстановитьБезопасныйРежим(Истина);
Выполнить Алгоритм;

В режиме сервиса, включение безопасного режима должно учитывать разделение данных, т.е. приведенный выше пример необходимо доработать следующим образом:

УстановитьБезопасныйРежим(Истина);

Для каждого ИмяРазделителя Из РазделителиКонфигурации() Цикл
 УстановитьБезопасныйРежимРазделенияДанных(ИмяРазделителя, Истина);
КонецЦикла;

Выполнить Алгоритм;

При использовании в конфигурации **Библиотеки стандартных подсистем**, следует использовать:

- ОбщегоНазначения.ВыполнитьВБезопасномРежиме() вместо **Выполнить**;
- ОбщегоНазначения.ВычислитьВБезопасномРежиме() вместо **Вычислить**;
- ОбщегоНазначения.ВыполнитьМетодКонфигурации() вместо формирования строки вызова метода модуля и передачи ее в **Выполнить**;
- ОбщегоНазначения.ВыполнитьМетодОбъекта() вместо формирования строки вызова метода объекта и передачи ее в **Выполнить**.

При использовании в конфигурации **Библиотеки стандартных подсистем** версии, меньшей чем 2.4.1, следует использовать:

- РаботаВБезопасномРежиме.ВыполнитьВБезопасномРежиме() вместо **Выполнить**;
- РаботаВБезопасномРежиме.ВычислитьВБезопасномРежиме() вместо **Вычислить**;
- РаботаВБезопасномРежиме.ВыполнитьМетодКонфигурации() вместо формирования строки вызова метода модуля и передачи ее в **Выполнить**;
- РаботаВБезопасномРежиме.ВыполнитьМетодОбъекта() вместо формирования строки вызова метода объекта и передачи ее в **Выполнить**.

3. Если произвольный код не может быть успешно выполнен в безопасном режиме (например, в нем используется обращение к файлам), то такой код должен предварительно пройти аудит и должен быть размещен в справочнике, к которому есть доступ только у пользователя ответственного за безопасность (например, администратора). Предлагается следующий сценарий работы с внешним кодом:

- внешний код рекомендуется поместить во внешнюю обработку или отчет (в крайнем случае, допустимо передать его, как фрагмент текста);
- передать внешний код администратору с пометкой, что необходимо выполнение кода в небезопасном режиме;
- администратор или сотрудник с соответствующей квалификацией должен выполнить аудит полученного кода;
- если код безопасен и надежен, администратор поместит внешний код в специальный справочник, доступ на запись в который есть только у него;
- вместо **Выполнить** или **Вычислить** следует (или):
 - подключить проверенную обработку из справочника и вызвать необходимый экспортный метод;
 - вызвать фрагмент кода с помощью специальной процедуры или функции, которая централизованно в конфигурации выполняет внешний код.

При использовании в конфигурации **Библиотеки стандартных подсистем**, для этих целей следует воспользоваться подсистемой **Дополнительные отчеты и обработки**.

4. Если конфигурация рассчитана на работу в модели сервиса, и в конфигурации предусмотрен перенос данных в сервис из локальной версии программы, необходимо обеспечить отключение всех пользовательских фрагментов кода или текстов запросов, которые были введены в локальной версии.

При использовании в конфигурации **Библиотеки стандартных подсистем**, также имеется возможность предварительной обработки данных, загружаемых из локальной версии в сервис (см. документацию к подсистеме **Работа в модели сервиса**).

См. также

- [Ограничения на выполнение «внешнего» кода](#)
- [Облачные технологии](#) (статья на сайте 1c.ru)
- [Безопасность прикладного программного интерфейса сервера](#)

Безопасность запуска приложений

Область применения: управляемое приложение, обычное приложение.

#std774

1. При запуске внешней программы из кода требуется составлять строку запуска таким образом, чтобы она собиралась только из проверенных частей.

Если одна из частей, из которых собирается строка запуска, содержит данные, полученные из базы данных, из поля ввода на форме или прочитаны из хранилища настроек, то перед запуском программы требуется проверить, являются ли запуск безопасным. Безопасными считаются такие строковые данные, которые не содержат в себе следующие символы: "\$", "``", "!", "||", ":", "&", "&&".

Данное требование распространяется на все способы запуска программы, в том числе:

- КомандаСистемы(<СтрокаКоманды>, <ТекущийКаталог>)
- ЗапуститьПриложение(<СтрокаКоманды>, <ТекущийКаталог>, <ДождатьсяЗавершения>, <КодВозврата>);
- НачатьЗапускПриложения(<ОписаниеОповещения>, <СтрокаКоманды>, <ТекущийКаталог>, <ДождатьсяЗавершения>);
- ПерейтиПоНавигационнойСсылке(<НавигационнаяСсылка>);
- Использование COM объектов "Wscript.Shell" и "Shell.Application".

2. При использовании **Библиотеки стандартных подсистем** для запуска внешних программ требуется использовать следующий программный интерфейс:

2.1. Для того чтобы открыть проводник с фокусировкой на указанном файле, использовать процедуру **ФайловаяСистемаКлиент.ОткрытьПроводник**.

Например:

```
// Для Windows  
ФайловаяСистемаКлиент.ОткрытьПроводник("C:\Users");  
ФайловаяСистемаКлиент.ОткрытьПроводник("C:\Program Files\1cv8\common\1cestart.exe");  
  
// Для Linux  
ФайловаяСистемаКлиент.ОткрытьПроводник("/home/");  
ФайловаяСистемаКлиент.ОткрытьПроводник("/opt/1c/v8.3/x86_64/1cv8c");
```

2.2. Для того чтобы открыть файл в программе просмотра, ассоциированной с расширением файла, использовать процедуру **ФайловаяСистемаКлиент.ОткрытьФайл**. Она исключает запуск исполняемых файлов (например, *.exe, *.bin, *.apk).

Например:

```
ФайловаяСистемаКлиент.ОткрытьФайл(КаталогДокументов() + "test.pdf");  
ФайловаяСистемаКлиент.ОткрытьФайл("D:\test.xlsx");
```

2.3. Для того чтобы открыть веб-страницу в браузере, запустить программу по протоколу (например, mailto:, skype:, tel: и.т.д) или открыть навигационную ссылку информационной базы следует использовать процедуру **ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку**. При этом в веб-клиенте пользователю будет предложено установить расширение для работы с файлами в тех случаях, когда оно необходимо для выполнения операции.

Например:

```
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("https://1c.ru");  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("e1cib/navigationpoint/startpage"); // начальная страница.  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("mailto:help@1c.ru");  
ФайловаяСистемаКлиент.ОткрытьНавигационнуюСсылку("skype:echo123?call");
```

В то же время, для открытия проводника или файла в программе просмотра не следует формировать ссылку по протоколу file://, для этого следует использовать одну из процедур: **ОткрытьПроводник** (см. п. 2.1) или **ОткрытьФайл** (см. п. 2.2).

2.4. Для того чтобы:

- запускать файлы на исполнение (например, *.exe, *.bat),
- использовать системные команды (например, ping, tracert или traceroute, обращаться к гас-клиенту),
- выполнять команды на сервере,
- а также получать код возврата и значения потоков вывода (stdout) и ошибок (stderr)

следует использовать **ФайловаяСистемаКлиент.ЗапуститьПрограмму** (в клиентском коде) и **ФайловаяСистема.ЗапуститьПрограмму** (в серверном коде).

Например:

```
ФайловаяСистемаКлиент.ЗапуститьПрограмму("calc");
```

Пример запуска с ожиданием завершения и получения кода возврата:

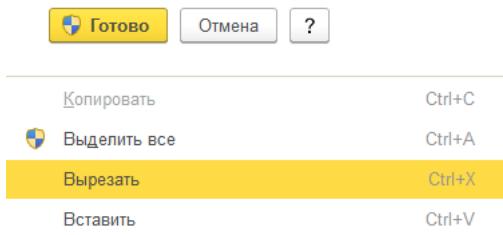
```
ПараметрыЗапускаПрограммы = ФайловаяСистема.ПараметрыЗапускаПрограммы();  
ПараметрыЗапускаПрограммы.ДождатьсяЗавершения = Истина;  
ПараметрыЗапускаПрограммы.ПолучитьПотокВывода = Истина;  
ПараметрыЗапускаПрограммы.ПолучитьПотокОшибок = Истина;
```

```
Результат = ФайловаяСистема.ЗапуститьПрограмму(  
    "ping 127.0.0.1 -n 5", ПараметрыЗапускаПрограммы);
```

```
КодВозврата = Результат.КодВозврата;  
ПотокВывода = Результат.ПотокВывода;  
ПотокОшибок = Результат.ПотокОшибок;
```

3. Для выполнения команды, требующей запуск внешней программы в режиме наивысших прав (например, в ОС Windows – с отображением запроса повышения прав UAC), необходимо:

- реализовывать ее на управляемой форме в виде кнопки или пункта меню;
- а на самой кнопке, начинающей выполнение действия, отобразить значок щита (общая картинка **ЗначокЩита** из **Библиотеки стандартных подсистем**).



См. например, [требования для ОС Windows](#).

Безопасность программного обеспечения, вызываемого через открытые интерфейсы

Область применения: управляемое приложение, обычное приложение.

#std775

1. При использовании интеграции со сторонними приложениями с помощью открытых интерфейсов (в частности, с помощью СОМ) требуется отключать исполнение произвольного кода средствами вызываемого приложения.

2. В частности, перед программным открытием документов Microsoft Word и Microsoft Excel через СОМ следует запрещать исполнение макросов. Иначе это может привести к выполнению вредоносных макросов (вирусов), если таковые присутствуют в документе.

НЕПРАВИЛЬНО открывать документ Microsoft Word по умолчанию:

```
ОбъектWord = Новый СОМОбъект("Word.Application");  
Документ = ОбъектWord.Documents.Open(ИмяФайла);
```

ПРАВИЛЬНО открывать документ Microsoft Word с отключением макросов:

```
ОбъектWord = Новый СОМОбъект("Word.Application");  
ОбъектWord.WordBasic.DisableAutoMacros(1);  
Документ = ОбъектWord.Documents.Open(ИмяФайла);
```

ПРАВИЛЬНО открывать документ Microsoft Excel с отключением макросов:

```
ОбъектExcel = Новый СОМОбъект("Excel.Application");
ОбъектExcel.AutomationSecurity = 3; // msoAutomationSecurityForceDisable = 3
Документ = ОбъектExcel.Workbooks.Open(Имяфайла);
```

3. Если при программном открытии документов Microsoft Office все же необходимо разрешить выполнение автоматически запускающихся макросов, тогда следует реализовать следующую логику работы:

- В конфигурации сделать формы настройки безопасности запуска макросов, отдельно для клиентского и серверного кода со следующими элементами:
 - о «Запретить автоматический запуск»;
 - о «Разрешить запуск подписанных макросов (рекомендуется)» (выбран по умолчанию);
 - о «Разрешить запуск не подписанных макросов (опасно)».
- Форма настройки для клиентского кода должна быть доступна каждому пользователю, настройки должны сохраняться в разрезе пользователей, каждому пользователю должны быть доступны только свои настройки.
- Форма настройки для серверного кода должна быть доступна только администратору, доступ к настройкам должен быть только у администратора.
- При программном открытии документов следует учитывать эти настройки.

3.1. Проверку наличия подписи макросов в документах Microsoft Word можно реализовать так:

```
ОбъектWord = Новый СОМОбъект("Word.Application");
ОбъектWord.WordBasic.DisableAutoMacros(1); // Отключить автозапуск
Документ = ОбъектWord.Documents.Open(Файлдокумента);
```

```
Если Документ.VBASigned Тогда
    ОбъектWord.WordBasic.DisableAutoMacros(0); // Включить автозапуск
    Документ.RunAutoMacro(2); // wdAutoOpen = 2
Иначе
    Документ.Close();
    ВызватьИсключение НСтр("ru = 'Документ не подписан. Открытие отменено.'");
КонецЕсли;
```

3.2. Проверку наличия подписи макросов в документах Microsoft Excel можно реализовать так:

```
ОбъектExcel = Новый СОМОбъект("Excel.Application");
ИсходныйУровеньБезопасности = ОбъектExcel.AutomationSecurity;
ОбъектExcel.AutomationSecurity = 3; // msoAutomationSecurityForceDisable = 3
Документ = ОбъектExcel.Workbooks.Open(Файлдокумента);
ОбъектExcel.AutomationSecurity = ИсходныйУровеньБезопасности;
```

```
Если Документ.VBASigned Тогда
    Документ.Close();
    Документ = ОбъектExcel.Workbooks.Open(Файлдокумента);
Иначе
    Документ.Close();
    ВызватьИсключение НСтр("ru = 'Документ не подписан. Открытие отменено.'");
КонецЕсли;
```

См. также: [О параметрах исполнения макросов в Microsoft Excel](#)

Настройка ролей и прав доступа

Область применения: управляемое приложение, обычное приложение.

#std689

Действует для конфигураций УП (ERP), УТ 11 и входящих в них библиотек.

Для других конфигураций рекомендуется к использованию.

Содержит уточнения к требованиям других стандартов.

1. Общие положения

1.1. Роли создаются «атомарными», т.е. дающими права на доступ к элементарным функциям программы. Из этих элементарных ролей создаются профили пользователей, которые уже и назначаются пользователям средствами БСП. Деление прав на доступ к объектам между функциями должно быть таким, чтобы на типовом внедрении не возникало необходимости в создании новых ролей.

Исключение: для ролей, назначаемых внешним пользователям, задается исчерпывающий набор прав к необходимым объектам.

Например, в УП(ERP) это роль **ПартнерСамообслуживание**.

1.2. Роль **ПолныеПрава** (англ. **FullAccess**) совместно с ролью **АдминистраторСистемы** (англ. **SystemAdministrator**) дает неограниченный доступ (без RLS) ко всем объектам. См. [стандартные роли](#).

1.3. Ни одна роль (в т.ч. **ПолныеПрава** и **АдминистраторСистемы**) не должна давать право на интерактивное удаление ссылочных объектов.

- после создания нового объекта нужно зайти в роль **ПолныеПрава** и отключить право интерактивного удаления у ссылочных объектов.

1.4. Только роль **ПолныеПрава** и **АдминистраторСистемы** должна давать право на удаление ссылочных объектов.

1.5. Только для роли **ПолныеПрава** должен быть установлен флаг «Устанавливать права для новых объектов». Для всех остальных ролей этот флаг должен быть снят.

1.6. Если какое-то право может быть использовано только администратором системы (например, использование какого-то отчета или обработки), то достаточно, чтобы это предоставлялось одной из ролей **ПолныеПрава** и **АдминистраторСистемы**, создавать отдельные роли в этом случае не требуется.

1.7. Во всех документах, предполагающих проведение, должны быть выставлены флаги «Привилегированный режим при проведении» и «Привилегированный режим при отмене проведения», поэтому не нужно создавать роли, дающие права на изменение регистров, подчиненных регистраторам.

Исключение: документы, предназначенные для непосредственной корректировки записей регистров, могут проводиться с проверкой прав доступа, но в этом случае необходимо предусмотреть роли, дающие права на изменение регистров.

1.8. Во всех функциональных опциях должны быть выставлены флаги «Привилегированный режим при получении».

Исключение: в конфигурации могут быть предусмотрены параметризованные ФО, для которых разработчик специально предусматривает различия в получаемых значениях пользователями с разными правами.

Пример: Есть параметризованная ФО **ИспользоватьВалютуПриРасчетеСПерсоналом**, которая параметризуется организацией. Если пользователь будет получать ее значение в контексте своих прав, то он не увидит поле «валюта» в документе, если у него нет ни одной организации, где применяется валютный учет.

1.9. Не должно быть ролей, кроме [стандартных ролей БСП](#), которые дают общие права (такие как **Администрирование**, **ТонкийКлиент** и т.п.).

- при создании новой роли нужно следить, чтобы эти права были выключены.

1.10. В отдельных случаях для неконфиденциальных данных и общедоступных функций не требуется создавать отдельную роль на чтение (а также просмотр и ввод по строке - для ссылочных данных), а следует включать эти права в роли **БазовыеПрава<ИмяБиблиотеки>** (англ. **BasicAccess<LibraryName>**) и **БазовыеПраваВнешнихПользователей<ИмяБиблиотеки>** (англ. **BasicAccessExternalUser<LibraryName>**) (эта роль необходима только если в конфигурации предусмотрена работа с **внешними пользователями**). Например, это константы, общенациональные классификаторы, общие формы выбора периода, ввода контактной информации и др.

1.11. Не допускается, чтобы одна роль давала права на объекты, которые относятся к другим **подсистемам**.

Например, в хранилище УП (ERP) нельзя, чтобы одна роль давала права на объекты, которые есть в УТ 11 и на объекты, которых в УТ 11 нет. См. также: [Разработка ролей в библиотеках](#).

2. Правила создания ролей к элементарным функциям

2.1. Объекты, при проектировании прав доступа, необходимо объединить в элементарные функции. Если объекты входят в одну функцию, то это означает, что не может быть задачи, когда доступ к этим объектам может быть разный.

Пример:
Есть документ «Заказ клиента» и связанный с ним регистр накопления «Заказы клиентов», который хранит остатки неотгруженных товаров и неоплаченных сумм. По сути этот регистр является отражением текущего состояния табличной части заказа. Если пользователь имеет права на документ, то будет странно, что он не будет иметь прав на регистр. Поэтому документ «Заказ клиента» и регистр накопления «Заказы клиентов» можно объединить в одну элементарную функцию. Если есть отчет, отображающий в удобном виде остатки регистра «Заказы клиентов», то логично его тоже включить в эту функцию.

Противоположный пример:

Есть документ «Реализация товаров и услуг», выступающий в роли распоряжения на отгрузку товаров. Остатки по распоряжениям на отгрузку товаров хранят регистр накопления «Товары к отгрузке». Объединять «Реализацию товаров и услуг» и регистр «Товары к отгрузке» в рамках одной функции было бы не правильно, т.к., например, кладовщик, вполне может иметь права на чтение регистра «Товары к отгрузке», но может не иметь прав на чтение документа «Реализация товаров и услуг».

2.2. В случае если возникают сомнения в том, что два объекта могут быть отнесены к одной элементарной функции, лучше выделить их в разные.

2.3. Каждый объект должен быть отнесен к элементарной функции, и только к одной.

2.4. Объекты, относящиеся к разным библиотекам не могут быть отнесены к одной элементарной функции.

3. Ссылочные объекты и регистры

3.1. Для функций, включающих в себя ссылочные объекты и независимые регистры сведений, должно быть создано две роли

- Чтение<ИмяФункции> (англ. **Read<FeatureName>**);
- ДобавлениеИзменение<ИмяФункции> (англ. **InsertUpdate<FeatureName>**) (или Изменение<ИмяФункции> (англ. **Update<FeatureName>**), если добавление выполняется автоматически, либо только администратором).

Роли должны содержать следующие права (когда они имеются у объекта метаданных):

- Чтение<ИмяФункции> содержит права:
 - Чтение, Просмотр, Ввод по строке.
- Изменение<ИмяФункции> содержит те же права, что и роль Чтение<ИмяФункции>, а также:
 - Изменение, Редактирование;
 - Проведение, Отмена проведения, Интерактивное проведение, Интерактивная отмена проведения, Интерактивное изменение проведенных (для документов);
 - Управление итогами (для регистров).
- ДобавлениеИзменение<ИмяФункции> содержит те же права, что и роль Изменение<ИмяФункции>, а также:
 - Добавление, Интерактивное добавление;
 - Интерактивная пометка удаления, Интерактивное снятие пометки удаления.

3.2. Необходимо помнить, что для регистров, подчиненных регистратору, обычно не требуется назначать права на изменение (см. п. 1.7).

4. Журналы документов

4.1. Если все документы, входящие в журнал, отнесены к одной элементарной функции, то права на чтение и просмотр журнала нужно включить во все роли этой функции.

4.2. Платформа **1С:Предприятие** имеет следующую особенность: если у пользователя нет прав на чтение любого документа, входящего в журнал документов, то платформа не дает доступ ко всем графикам журнала, в котором отражается этот документ, для всех документов, входящих в журнал. Поэтому нет никакого практического смысла создавать отдельную роль для доступа к журналу: журнал может входить в элементарную функцию, или может быть доступен только пользователю с полными правами.

5. Константы

5.1. Если предполагается, что константа должна изменяться только администратором системы, то права на изменение и редактирование должны быть только у одной из ролей: **ПолныеПрава** или **АдминистраторСистемы**. Эти роли должны включать также и права на чтение и просмотр констант.

5.2. Если предполагается, что константу может менять пользователь, то нужно включить права на чтение, просмотр, изменение и редактирование в уже имеющуюся «настроечную» роль или создать отдельную роль **Изменение<ИмяКонстанты>** (англ. **Update<ConstantName>**), дающую права на чтение, просмотр, изменение и редактирование этой константы.

5.3. В подавляющем большинстве случаев константа используется для хранения неконфиденциальной информации, поэтому права на чтение и просмотр константы нужно назначать роли **БазовыеПрава<ИмяБиблиотеки>** и **БазовыеПраваВнешнихПользователей<ИмяБиблиотеки>**. Это позволяет избежать неоправданной установки **привилегированного режима** при чтении значения константы из кода.

5.4. В редких случаях, когда константа используется для хранения конфиденциальной информации, необходимо создать роль **Чтение<ИмяКонстанты>** (англ. **Read<ConstantName>**). При этом, если значение должно быть доступно только администратору, создавать отдельную роль на чтение не требуется.

Например, для константы **ПараметрыАдминистрированияИБ** создавать отдельную роль на чтение не требуется – достаточно включения прав на чтение и просмотр в роль **АдминистраторСистемы**.

6. Подсистемы, отображаемые в главном командном интерфейсе

6.1. Для каждой подсистемы верхнего уровня должна быть создана роль **Подсистема<ИмяПодсистемы>** (англ. **Subsystem<SubsystemName>**), дающая право на просмотр

6.2. Если интерфейс подсистемы организован так, что часть настроек и справочников отображаются в отдельной форме, то роль, дающая право на подсистему, должна включать права на просмотр этой формы (например, в УП(ERP) часть справочников в разделах командного интерфейса не вынесены и отображаются в форме, вызываемой командой «Настройки и справочники»).

7. Отчеты

7.1. Если отчет строится на основе данных, входящих в одну элементарную функцию (п. 2.1), то в общем случае права на доступ к такому отчету можно включить в роли, созданные по этой элементарной функции.

Пример:
Отчет по исполнению заказов клиентов полностью строится на основе данных регистра накопления ЗаказыКлиентов, поэтому права на отчет нужно включить в роль, дающую право на чтение регистра.

7.2. Если на внедрении могут возникнуть задачи отдельной настройки прав к отчету, который строится на основе данных, входящих в одну элементарную функцию, то для доступа к такому отчету необходимо создать отдельную роль **ПросмотрОтчета<ИмяОтчета>** (англ. **ViewReport<ReportName>**), дающую права использования и просмотра.

Пример:
Хотя отчет по контактной информации отображает данные входящие в элементарную функцию информации о клиентах, но на внедрении может потребоваться ограничить доступ к отчету, который позволяет вывести информацию по всем клиентам сразу.

7.3. Если отчет строится на основе данных, входящих в несколько элементарных функций, необходимо создать роль **ПросмотрОтчета<ИмяОтчета>**, дающую права использования и просмотра.

Пример:
Отчет по выполнению плана продаж строится на основе данных о планах и данных о продажах. Права чтение этих данных дают роли, относящиеся к разным элементарным функциям. Для реализации доступа к отчету нужно создать отдельную роль.

7.4. Однотипные отчеты при проектировании прав доступа можно объединить в элементарные функции при соблюдении следующих условий:

- отчеты не входят в другие элементарные функции (см. п. 7.1);
- на внедрении не может возникнуть задачи различной настройки прав доступа к этим отчетам.

8. Обработки и общие формы

8.1. Для каждой обработки, представляющей из себя рабочее место, т.е. в ГКИ есть отдельная команда для открытия этой обработки, должна быть создана роль **ИспользованиеОбработки<ИмяОбработки>** (англ. **Use DataProcessor<DataProcessorName>**), дающая права на просмотр и использование.

При этом не допускается объединять права доступа к разным обработкам-рабочим местам в одной роли.

8.2. Права ко всем другим типам обработок, например

- вспомогательные обработки, которые нельзя вызывать из глобального командного интерфейса, а можно открыть только из других объектов, например, подбор товаров;
- обработки, в которых расположен общий код, например, код формирования печатных форм;

должны быть назначены в роли **БазовыеПрава<ИмяБиблиотеки>**.

8.3. Права к обработкам, предназначенным исключительно для администратора программы, нужно назначать только роли **ПолныеПрава**, не создавая отдельных ролей для таких обработок.

8.4. Все эти правила равным образом применяются для общих форм. Название роли для общей формы, описанной в п. 8.1. – **ИспользованиеОбщейФормы<ИмяОбщейФормы>** (англ. **UseCommonForm<CommonFormName>**).

8.5 Исключения из этих правил описаны в п. 6.2

Пример:

В УТ 11 права к обработкам **ПодборТоваров** и **ПоискОбъектовПоШтрихкоду** назначены роли **БазовыеПраваУТ**.

9. Команды

9.1. Если команда не предполагает изменение данных, то в общем случае право на просмотр должно быть назначено той роли, которая дает право на просмотр объекта.

- т.к. роль, дающая право на изменение объекта дает также права на чтение объекта, нужно не забывать приставлять права на команду и в роли с правом на чтение, и в роли с правом на изменение;
- права на команды печати, которые расположены в обработках (это печатные формы, которые печатаются из нескольких объектов) нужно назначать роли **БазовыеПрава<ИмяБиблиотеки>**

9.2. Если команда связана с изменением данных, то право на просмотр нужно назначить роли, которая дает права на изменение объектов.

10. Права, не связанные с доступом к объектам

10.1. В случае если возникает необходимость давать пользователям какие-то дополнительные права, не связанные с доступом к объектам, нужно создавать роль **<НаименованиеПрава>**, не дающую доступ ни к одному объекту. При этом в наименовании не нужно использовать слово «Право».

Пример:

Правильно **ОтклонениеОтУсловийЗакупок**

Неправильно **ПравоСозданияВыпускаПродукцииБезЗаказа**

10.2. В коде конфигурации нужно проверять наличие у пользователя этой роли. Пользователь с ролью **ПолныеПрава** должен проходить проверку без необходимости включения в его профиль этой роли. Для проверки следует использовать **функцию БСП Пользователи.РолиДоступны**.

10.3. Использование других механизмов, кроме проверки наличия роли (или какого-то права) при реализации дополнительных прав пользователя, не допускается.

11. Права для внешних пользователей

Роли, предназначенные исключительно для предоставления прав доступа **внешним пользователям** (представленным в программе одним из объектов, например, элементами справочников **Сотрудники**, **Партнеры**, **Физические лица** и др.), следует называть с определенным префиксом.

Например, префикс **Самообслуживание** для доступа к рабочему месту по самообслуживанию клиентов в торговом решении:

- **СамообслуживаниеОформлениеПретензий**, синоним **Самообслуживание: оформление претензий**
- **СамообслуживаниеПросмотрОтчетаСостоянияВыполненияЗаказа**, синоним **Самообслуживание: просмотр отчета «Состояние выполнения заказа»**
- **СамообслуживаниеДобавлениеИзменениеКонтрагентов**, синоним **Самообслуживание: добавление и изменение контрагентов**

12. Права к устаревшим объектам

Устаревшие объекты метаданных с префиксом **Удалить** должны быть исключены из всех ролей, кроме ролей **ПолныеПрава** или **АдминистраторСистемы**.

См. также

- [Стандартные роли](#)
- [Разработка ролей в библиотеках](#)
- [Эффективные условия запросов \(пункт 7\)](#)

Стандартные роли

Область применения: управляемое приложение, обычное приложение.

#std488

1.1. Если в конфигурации есть разграничение прав доступа пользователей к данным информационной базы, то настройку доступа следует выполнять с помощью ролей. Роли должны создаваться разработчиком конфигурации исходя из задачи ограничения доступа пользователей к данным.

1.2. В простейшем случае, роли в конфигурации могут непосредственно соответствовать должностным обязанностям пользователей (**Директор**, **Бухгалтер**, **Кладовщик** и т.п.).

Для возможности более тонкой настройки прав доступа администратором, роли могут соответствовать более «мелким» отдельным функциям, совокупность которых образует набор прав для конкретной категории пользователей. В этом случае пользователям назначается комбинация ролей (например, **ЧтениеПроизводственныхДокументов**, **ДобавлениеИзменениеСкладскихДокументов**, **ЧтениеНСИ** и т.п.). Для упрощения администрирования рекомендуется поставлять в составе конфигурации типовые комбинации таких ролей (например, готовый профиль для директора, бухгалтера, кладовщика и т.п.) При использовании в конфигурации **Библиотеки стандартных подсистем** для этого можно воспользоваться средствами подсистемы **«Управление доступом»**.

2. В конфигурации должны быть определены три обязательные роли, которые предназначены для «прикладного» и системного администрирования информационной базы, а также для интерактивного открытия внешних отчетов и обработок:
ПолныеПрава (синоним «Полные права»), **АдминистраторСистемы** (синоним «Администратор системы») и **ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок** (синоним «Интерактивное открытие внешних отчетов и обработок»).

2.1. ПолныеПрава (англ. **FullAccess**) - обязательная роль, которая предоставляет неограниченный доступ ко всем «прикладным» данным информационной базы, но не дает прав доступа для администрирования информационной базы в целом (обновление конфигурации, работа в конфигураторе и т.п.).
Эта роль должна:

- позволять самостоятельное использование (может быть назначена пользователям);
- предоставлять неограниченный доступ ко всем данным области (к разделенным данным), кроме права интерактивного удаления (см. также п.5);
- позволять выполнять все административные действия с областью данных (администрирование пользователей, настройка программы, удаление помеченных объектов и т.п.);
- включать в себя перечисленные права:
 - Администрирование данных
 - Активные пользователи
 - Журнал регистрации
 - Монопольный режим
 - Тонкий клиент
 - Веб-клиент
 - Сохранение данных пользователя
 - Вывод

В случае если конфигурация рассчитана на работу в модели сервиса, то роль **ПолныеПрава** назначается администраторам абонентов (администраторам областей данных).

При работе конфигурации в локальном режиме роль **ПолныеПрава** назначается пользователям совместно с ролью **АдминистраторСистемы**, так как в этом режиме функции системного и «прикладного» администрирования информационной базы, как правило, совмещены.

2.2. АдминистраторСистемы (англ. **SystemAdministrator**) – обязательная роль, предоставляющая дополнительные права на администрирование информационной базы в целом (обновление конфигурации, работа в конфигураторе и т.п.).
Эта роль должна:

- назначаться пользователям только совместно с ролью **ПолныеПрава**;
- предоставлять неограниченный доступ ко всем неразделенным данным информационной базы;
- содержать все права доступа к объектам (кроме права интерактивного удаления - см. ниже п.5);
- включать в себя все права к корню конфигурации (в частности, права **Администрирование** и **Администрирование данных**), кроме прав **Интерактивное открытие внешних отчетов** и **Интерактивное открытие внешних обработок**.

2.3. ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок (англ. **InteractiveOpenExternalReportsAndDataProcessors**) обязательная роль, предоставляющая дополнительные права на открытие внешних отчетов и обработок через меню «Файл – Открыть».
Эта роль должна:

- включать в себя права к корню конфигурации **Интерактивное открытие внешних отчетов** и **Интерактивное открытие внешних обработок**.

При работе конфигурации в локальном режиме роль **ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок** назначается администраторам, но в целях повышения безопасности информационной базы администратор может запретить использование данной роли.

При работе в модели сервиса роли **АдминистраторСистемы**, **ПолныеПрава** и **ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок** назначаются администраторам сервиса.

2.4. Роли ПолныеПрава, АдминистраторСистемы и ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок должны устанавливаться как основные роли конфигурации (свойство **ОсновныеРоли**).

Методическая рекомендация (полезный совет)

2.5. При необходимости дать возможность удаления объектов «неполноправным» пользователям, рекомендуется добавить отдельную роль **УдалениеПомеченныхОбъектов** (англ. **DeleteMarkedObjects**). Такая роль не предназначена для самостоятельного использования, ее следует назначать пользователям совместно с остальными ролями конфигурации.

3. Роли для настройки общих прав на информационную базу. В случае если в конфигурации для пользователей необходимо настраивать общие права работы с информационной базой (такие как «Тонкий клиент», «Толстый клиент», «Интерактивное открытие внешних обработок» и т.д.), то в конфигурации должны быть определены отдельные роли для предоставления этих прав. Такие роли не предназначены для самостоятельного использования, их следует назначать пользователям совместно с остальными ролями конфигурации.

Конфигурация должна быть одинаково рассчитана на работу как при наличии этих ролей, так и в условиях отсутствия любой из этих ролей у пользователя.

3.1. Администрирование (англ. **Administration**) - предоставляет права «Администрирование», «Администрирование данных», «Администрирование расширений конфигурации» и «Активные пользователи».

3.2. ВыводНаПринтерФайлБуферОбмена (англ. **OutputToFileClipboard**) - предоставляет право «Вывод».

3.3. ЗапускAutomation (англ. **StartAutomation**) - предоставляет право «Automation».

3.4. ЗапускВебКлиента (англ. **StartWebClient**) - предоставляет право «Веб-клиент».

3.5. ЗапускВнешнегоСоединения (англ. **StartExternalConnection**) - предоставляет право «Внешнее соединение»

3.6. ЗапускТолстогоКлиента (англ. **StartThickClient**) - предоставляет право «Толстый клиент».

3.7. ЗапускТонкогоКлиента (англ. **StartThinClient**) - предоставляет право «Тонкий клиент».

3.8. ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок (англ. **InteractiveOpenExternalReportsAndDataProcessors**) - предоставляет права «Интерактивное открытие внешних обработок» и «Интерактивное открытие внешних отчетов».

3.9. ОбновлениеКонфигурацииБазыДанных (англ. **UpdateDatabaseConfiguration**) - предоставляет право «Обновление конфигурации базы данных».

3.10. ПросмотрЖурналаРегистрации (англ. **ViewEventLog**) - предоставляет право «Журнал регистрации».

3.11. РежимВсеФункции (англ. **AllFunctionsMode**) - предоставляет право «Режим "Все функции"». Этот режим предназначен только для разработчиков/внедренцев и для разбора нештатных ситуаций, поэтому конфигурация должна обеспечивать работу пользователей без использования этого режима. Например, все стандартные обработки (удаление помеченных, управление итогами и агрегатами и т.п.) должны быть доступны соответствующим пользователям в разделах интерфейса программы.

3.12. СохранениеДанныхПользователя (англ. **SaveUserData**) - предоставляет право «Сохранение данных пользователя». Рекомендуется предоставлять эту роль всем категориям пользователей, за редким исключением, когда требуется явно запретить настройку пользовательского интерфейса и любые другие персональные настройки таким образом, чтобы работа пользователя не оставляла никаких «следов» в информационной базе. Как правило, такой режим работы требуется для внешних или временных пользователей (респонденты, аудиторы и пр.) или для пользователей, работающих по тем или иным причинам под одной учетной записью.

Конфигурация должна быть рассчитана на работу пользователей без роли (права) **СохранениеДанныхПользователя**. В случае если конфигурация обращается из кода

- к пользовательским настройкам (сохранение и загрузка из различных хранилищ настроек: **ХранилищеОбщихНастроек**, **ХранилищеВариантовОтчетов**, **ХранилищеНастроекДанныхФорм**, **ХранилищеПользовательскихНастроекОтчетов**, **ХранилищеСистемныхНастроек**)
- к истории работы пользователя (**ИсторияРаботыПользователя**) и избранному (**ИзбранноеРаботыПользователя**)
- к пользовательским настройкам отчетов (метод **УстановитьТекущиеПользовательскиеНастройки** расширения управляемой формы для отчета)

то при отсутствии у пользователя права **СохранениеДанныхПользователя**, этот код должен быть пропущен таким образом, чтобы это не оказывало влияния на основные сценарии работы пользователя. Кроме того, если в конфигурации предусмотрены пользовательские интерфейсы или отдельные элементы форм для работы с пользовательскими настройками (история вводимых значений, флагки «Запомнить мой выбор» и пр.), то они не должны быть доступны пользователям без права **СохранениеДанныхПользователя**.

При использовании в конфигурации **Библиотеки стандартных подсистем** можно также воспользоваться функциями **ХранилищеОбщихНастроекЗагрузить** и **ХранилищеОбщихНастроекСохранить** общего модуля **ОбщегоНазначения**.

См. также: Работа с пользовательскими настройками

4.1. В тех случаях когда определенным пользователям требуется временный или постоянный доступ на просмотр всех данных информационной базы без ограничений, рекомендуется поставлять в составе конфигурации отдельные роли. Например, это может быть временный доступ для аудитора, постоянный – для собственника или директора организации.

4.2. В простейшем случае, когда роли в конфигурации соответствуют должностным обязанностям пользователей (Директор, Бухгалтер, Кладовщик и т.п.), должна быть добавлена отдельная роль **ТолькоПросмотр** (англ. *ViewOnly*).

Роль **ТолькоПросмотр** должна включать права **Чтение, Использование, Просмотр, Ввод по строке** (если применимо) для большинства объектов метаданных (за исключением тех данных, которые никогда не выводятся пользователю, а используются в конфигурации только в технологических целях, и поэтому доступ к ним осуществляется всегда в привилегированном режиме).

4.3. В конфигурациях, в которых роли соответствуют более «мелким» отдельным функциям (совокупность которых образует набор прав для конкретной категории пользователей), рекомендуется назначать пользователям комбинацию ролей, которые предоставляют доступ только на чтение (например, **ЧтениеПроизводственныхДокументов, ЧтениеКассовыхДокументов, ЧтениеКонтактнойИнформации** и т.п.). Рекомендуется поставлять в составе конфигурации типовые комбинации таких ролей (например, готовый профиль для аудитора, собственника, директора и т.п.)

При этом для неограниченного доступа на просмотр всех данных информационной базы для таких пользователей нужно отключать (не задавать) условия ограничения доступа на уровне записей (RLS).

5. Ни в одной роли, включая **ПолныеПрава** и **АдминистраторСистемы**, не должно быть установлено (кроме отдельных обоснованных случаев) следующих прав:

- Право интерактивного удаления
- Интерактивное удаление предопределенных данных
- Интерактивная пометка удаления предопределенных данных
- Интерактивное снятие пометки удаления предопределенных данных
- Интерактивное удаление помеченных предопределенных данных

Право удаления рекомендуется оставить только в ролях **ПолныеПрава** и **АдминистраторСистемы**.

См. также

- [Эффективные условия запросов \(пункт 7\)](#)

Установка прав для новых объектов и полей объектов

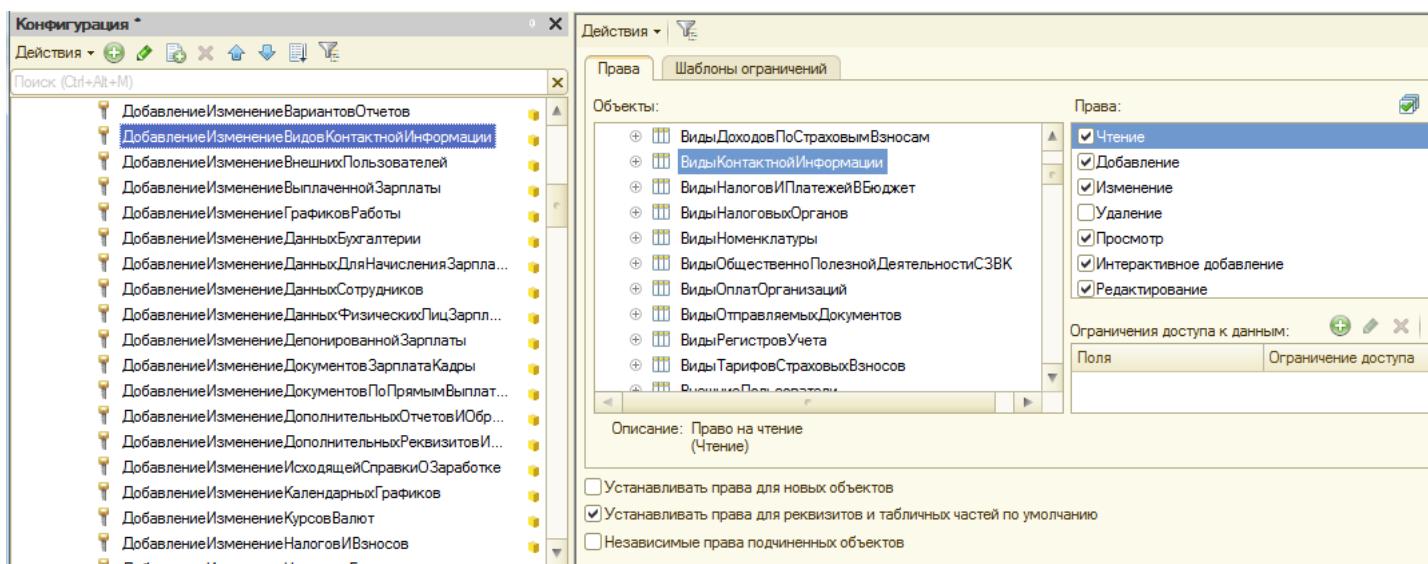
#std532

Область применения: управляемое приложение, обычное приложение.

При разработке ролей конфигурации следует придерживаться такой методики установки прав на объекты метаданных, которая не допускает появления в конфигурации ролей, дающих доступ к полям объекта, но не к самому объекту; и как следствие, позволяет избежать потенциальные проблемы настройки прав доступа на внедрении, когда пользователь может ошибочно получить доступ ко всем реквизитам объекта метаданных при назначении ему подобной роли.

1. Флажок «Устанавливать права для новых объектов» должен быть установлен только у роли **ПолныеПрава**
2. При добавлении новой роли, флажок «Устанавливать права для реквизитов и табличных частей по умолчанию» должен быть установлен, а флажок «Независимые права подчиненных объектов» должен быть снят.
3. При необходимости установить в роли права только на поля какого-либо объекта метаданных (права просмотр, редактирование на реквизиты, табличные части, измерения, команды и т.п. без этих прав на сам объект), предварительно требуется следующее. В роли установить флажок «Независимые права подчиненных объектов», а флажок «Устанавливать права для реквизитов и табличных частей по умолчанию» снять с очисткой прав на все реквизиты и табличные части.
4. При добавлении в конфигурацию новых объектов или новых полей существующих объектов следует настроить права на эти объекты или поля в соответствующих ролях.

Пример настройки прав в роли **ДобавлениеИзменениеВидовКонтактнойИнформации**:



Проверка прав доступа

Область применения: управляемое приложение, обычное приложение.

#std737

1. В случае большого количества ролей в конфигурации (от нескольких десятков) не рекомендуется использовать ролевую настройку видимости в элементах форм (просмотр и редактирование реквизитов по ролям, пользовательскую видимость полей формы по ролям, использование команд по ролям). Вместо этого следует придерживаться следующих подходов:

- при сильных различиях внешнего вида и функциональности формы в зависимости от наличия тех или иных ролей у пользователя – разрабатывать отдельные формы, специализированные под конкретный набор прав пользователя;
- при незначительных отличиях – выполнять проверку прав в коде. При этом следует иметь в виду, что программное управление видимостью может снизить скорость открытия формы, что нужно учитывать при выборе между предлагаемыми подходами.

Эти меры позволяют:

- существенно упростить работу из кода с элементами управления, которые пропадают из коллекции Элементы (код Элементы.ИмяЭлемента.ИмяСвойства становится нерабочим);
- повысить устойчивость кода к пересмотру состава ролей в конфигурации;
- организовать контроль использования ролей в конфигурации (в противном случае, выполнять анализ прав доступа по флажкам, назначенных для ролей в различных элементах производных форм конфигурации, крайне затруднительно).

2. Не рекомендуется использовать ролевую настройку видимости в командном интерфейсе конфигурации, командном интерфейсе основного раздела, а также рабочей области начальной страницы. Вместо этого следует использовать настройку прав на разделы командного интерфейса, общие формы и объекты, включенные в командный интерфейс или в рабочую область. Это

позволяет повысить предсказуемость поведения управляемого интерфейса для пользователя, а также упростить расследование ошибок.

3. Для проверки прав доступа в коде следует использовать метод **ПравоДоступа**.

Например, неправильно:

Если РольДоступна("ДобавлениеИзменениеСтранМира") Тогда ...
Если РольДоступна("ПросмотрОтчетаПопулярныеСтраны") Тогда ...

правильно:

Если ПравоДоступа("Редактирование", Метаданные.Справочники.СтраныМира) Тогда ...
Если ПравоДоступа("Просмотр", Метаданные.Отчеты.ПопулярныеСтраны) Тогда ...

Такой подход позволяет повысить устойчивость кода к пересмотру состава ролей в конфигурации.

4. В тех случаях, где роль не дает никаких прав на объекты метаданных, а служит только для определения того или иного дополнительного права, следует использовать метод **РольДоступна**.

При использовании в конфигурации **Библиотеки стандартных подсистем** (БСП) следует использовать функцию **РолиДоступны** общего модуля **Пользователи**:

Например, без использования БСП:

Если РольДоступна(...) Или <ЭтоПолноправныйПользователь> Или ПривилегированныйРежим() Тогда ...

Либо аналогичная проверка с использованием БСП:

Если Пользователи.РолиДоступны(...) Тогда ...

См. также

- [Стандартные роли](#)
- [Разработка ролей в библиотеках](#)

Использование привилегированного режима

Область применения: управляемое приложение, обычное приложение.

#std485

1.1. Привилегированный режим позволяет

- выполнить операции с данными от лица пользователей, которым данные недоступны;
- ускорить работу, так как в привилегированном режиме не накладываются ограничения на доступ к данным.

1.2. Привилегированный режим следует использовать

- когда требуется с логической точки зрения отключить проверку прав;
- когда допустимо отключить проверку прав, чтобы ускорить работу и при этом работа с данными от лица пользователя логически не нарушает установленные для него права доступа.

2. Далее приведем некоторые типовые сценарии использования привилегированного режима.

2.1. **Если подчиненные регистры (движения документов) требуются пользователю для получения отчетов**, то такие регистры следует оставлять доступными только для чтения, а запись в такие регистры следует выполнять переходом в привилегированный режим. При таком подходе, проверка прав на запись выполняется только для регистратора.

Например, в конфигурации имеются объекты метаданных:

- [Документ.ОприходованиеТоваров](#)
- [РегистрНакопления.ОстаткиНаСкладах](#)
- [Отчет.ОстаткиНаСкладах](#)

Документ **ОприходованиеТоваров** делает движения по регистру накопления **ОстаткиНаСкладах** при проведении в привилегированном режиме. Пользователю не требуется напрямую изменять регистр, но требуется формировать отчет **ОстаткиНаСкладах**, который читает данные регистра. Поэтому пользователю достаточно дать только право чтения регистра.

2.2. **Если регистры не требуются пользователю ни для получения отчетов, ни в командном интерфейсе**, то следует не давать пользователю права на их чтение. Чтение или запись данных в такие регистры выполняется переходом в привилегированный режим.

Например, в конфигурации имеется регистр сведений **ВерсииПодсистем**, данные которого не выводятся в отчетах, а доступ к нему осуществляется только через специальные функции общего модуля. В этом случае следует не давать пользователям какие-либо права доступа к регистру.

2.3. **Если для выполнения разрешенной пользователю операции требуются заранее недоступные данные**, тогда следует их получать переходом в привилегированный режим, но с гарантией предотвращения передачи этих данных на клиентскую сторону и с гарантией их использования на сервере только для выполнения запланированной операции.

Например, в случае если в конфигурации есть объекты метаданных

- [Документ.РеализацияТоваровИУслуг](#)
- [РегистрНакопления.Продажи](#)
- [РегистрНакопления.Взаиморасчеты](#)

и необходимо, чтобы

- проведение документа от лица кладовщика изменяло регистр продаж, но кладовщику регистр продаж оставался недоступным.
- при проведении документа от лица кладовщика проверялось состояние взаиморасчетов, но кладовщику регистр взаиморасчетов оставался недоступным.

2.4. Привилегированный режим также следует использовать с целью повышения производительности в тех случаях, когда это не приводит к выполнению действий или получению данных, не разрешенных пользователю.

Пример 1. Выполняется запрос, у которого значения параметров отбора уже прошли проверку ограничения прав на уровне записей. Тогда сам запрос допустимо выполнить в привилегированном режиме для повышения его производительности. Переход в привилегированный режим, например, при получении данных для отчета, следует тщательно рассчитывать, чтобы предотвратить получение данных, не разрешенных пользователю.

Пример 2. Параметр сеанса "Текущий пользователь" не доступен ни в одной роли. Для получения значения используется функция:

**Функция ТекущийПользователь() Экспорт
УстановитьПривилегированныйРежим(Истина);
Возврат ПараметрыСеанса.ТекущийПользователь;
КонецФункции**

3. В то же время, неоправданное использование привилегированного режима может привести к проблемам безопасности пользовательских данных.

3.1. Потенциально опасны любые экспортные процедуры и функции, которые выполняют на сервере какие-либо действия с предварительной безусловной установкой привилегированного режима, так как это отключает проверку прав доступа текущего пользователя. Особого внимания требуют экспортные процедуры и функции [клиентского прикладного программного интерфейса сервера 1С:Предприятия](#).

Например, неправильно:

Процедура ИзменитьИлиУдалитьДанные(...) Экспорт

```
УстановитьПривилегированныйРежим(Истина); // Отключаем проверку прав доступа
// Изменяем данные в привилегированном режиме
...
```

КонецПроцедуры

Правильно:

Процедура ИзменитьИлиУдалитьДанные(. . .) Экспорт

```
// Изменяем данные
// (при этом если у пользователя недостаточно прав для выполнения операции над данными, то будет вызвано исключение)
...
```

КонецПроцедуры

Исключение составляют случаи, когда действие, выполняемое процедурой, должно быть разрешено (или возвращаемое значение функции должно быть доступно) абсолютно всем категориям пользователей.

3.2. В случаях, когда к экспортной процедуре или функции выполняется обращение в сеансе с недостаточным уровнем прав доступа, должно вызываться исключение стандартного вида (см. метод платформы [ВыполнитьПроверкуПравДоступа](#)).

Как правило, для этого не требуется предусматривать в коде каких-либо проверок, так как при наличии в конфигурации [ролей и ограничений доступа к данным на уровне записей \(RLS\)](#) при попытке обращения к недоступным данным возникнет исключение:

- при выполнении запроса без [ключа слово РАЗРЕШЕННЫЕ](#);
- при вызове метода [ПолучитьОбъект](#) «от» недоступной ссылки на объект;
- при вызове недоступного метода глобального контекста;
- и в других аналогичных случаях.

При необходимости выполнить проверку прав доступа вручную, следует использовать метод [ВыполнитьПроверкуПравДоступа](#).

Пример предварительной проверки перед выполнением действий в привилегированном режиме:

Процедура ИзменитьИлиУдалитьДанные(. . .) Экспорт

```
ВыполнитьПроверкуПравДоступа( . . . ); // Если у пользователя недостаточно прав, то будет вызвано исключение
УстановитьПривилегированныйРежим(Истина); // Отключаем проверку прав доступа
```

```
// Изменяем данные в привилегированном режиме
...
```

КонецПроцедуры

См. также: [Перехват исключений в коде](#)

3.3. Включать привилегированный режим следует точечно, чтобы остальной код не терял возможности проверки прав пользователя.
Для этого нужно:

- установить привилегированный режим точно перед выполнением действия;
- выполнить действие без проверки прав;
- отключить привилегированный режим сразу же после выполнения действия;
- продолжить выполнение кода в непривилегированном режиме.

Например, неправильно:

Процедура ПодготовитьДанные()

```
УстановитьПривилегированныйРежим(Истина);
// Код подготовки запроса, которому требуется проверка прав
// Код выполнения запроса, для которого требуется отключить проверку прав
Выборка = Запрос.Выполнить().Выбрать();
// Обработка результатов запроса, где требуется проверка прав.
```

Правильно:

Процедура ПодготовитьДанные()

```
// Код подготовки запроса
УстановитьПривилегированныйРежим(Истина);
Выборка = Запрос.Выполнить().Выбрать();
УстановитьПривилегированныйРежим(Ложь);

// Обработка результатов запроса
```

4. Для перехода в привилегированный режим следует использовать следующие возможности платформы **1С:Предприятие**:

1. При реализации логики проведения и отмены проведения документов в обработчиках модуля документа **ОбработкаПроведения** и **ОбработкаУдаленияПроведения**:
 - с помощью установки свойств документа "Привилегированный режим при проведении" и "Привилегированный режим при отмене проведения" (эти свойства переводят выполнение обработчиков в привилегированный режим);
 - если логика проведения работает с некоторой выборкой данных, состав которой должен ограничиваться для пользователя, то следует снимать эти свойства и использовать процедуру глобального контекста **УстановитьПривилегированныйРежим**;
2. Устанавливать свойство бизнес-процесса "Привилегированный режим" для создания задач в привилегированном режиме.
3. Устанавливать свойство "Привилегированный режим при получении" для [функциональных опций](#).
4. В произвольном месте любого модуля с помощью процедуры глобального контекста **УстановитьПривилегированныйРежим**:

```
УстановитьПривилегированныйРежим(Истина)
// фрагмент кода в привилегированном режиме
// ...
УстановитьПривилегированныйРежим(Ложь)
```

4. С помощью размещения процедур и функций в общем модуле с установленным свойством **Привилегированный**.

Ограничения на использование ключевого слова "РАЗРЕШЕННЫЕ" в запросах

Область применения: управляемое приложение, обычное приложение.

#std415

Использование ключевого слова **РАЗРЕШЕННЫЕ** во всех запросах с целью нормальной работы пользователя без возникновения ситуаций с нарушением прав доступа, может привести к искажению данных, полученных в результате выполнения запроса. Необходимо иметь в виду, что для корректной работы бизнес-логики, заложенной в конфигурацию, различным подсистемам конфигурации необходимо предоставить доступ не к ограниченной, а ко всей требуемой информации.

Например, если использовать ключевое слово **РАЗРЕШЕННЫЕ** в запросах механизмов определения списания себестоимости товаров при их отгрузке со склада, то себестоимость списания одного пользователя может отличаться от себестоимости другого. В данном случае необходимо поддерживать целостность информации и либо разрешать пользователю чтение информации для корректного отражения хозяйственной операции, либо конфигурация должна прекращать выполнение операции с сообщением о нарушении прав доступа.

Другой пример: если у пользователя отсутствует доступ к контактной информации контактных лиц контрагентов, то пользователь ее не увидит. Данная информация не задействована в бизнес-процессах заложенных в конфигурацию. Ограничение доступа к ней не повлияет на работу конфигурации в целом.

Влияние изменения значений параметров сеанса и функциональных опций на производительность механизма ограничения доступа к данным

Область применения: управляемое приложение, обычное приложение.

#std491

В процессе работы система формирует специальный кеш запросов ограничений доступа к данным, создаваемый для того, чтобы повысить производительность запросов получения данных при использовании ограничений. Ограничения доступа к данным могут использовать параметры сеанса и функциональные опции. В кеш попадают запросы ограничения доступа с установленными значениями параметров сеанса и функциональных опций.

Однако, при изменении значений параметров сеанса или функциональных опций, которые используются в запросах ограничения доступа к данным, происходит очистка накопленного кеша запросов, что приводит к существенному снижению производительности запросов к данным.

Поэтому рекомендуется выполнять установку значений параметров сеанса "[по требованию](#)", в обработчике **Установка Параметров Сеанса** модуля сеанса. Также не рекомендуется часто выполнять изменение значений функциональных опций в процессе работы системы.

См. также

- [Использование функциональных опций](#)
- [Установка параметров сеанса "по требованию"](#)

Настройка обмена данными для классификаторов между различными информационными базами

Область применения: управляемое приложение, мобильное приложение.

#std637

1. К классификаторам относятся справочники и регистры сведений, которые содержат условно-постоянную информацию, одинаковую для всех совместно используемых информационных баз. Примерами классификаторов являются классификаторы банков, стран мира, валют, общероссийский классификатор единиц измерения (ОКЕИ), адресный классификатор и пр.

При использовании инструмента «Конвертация данных» для создания обменов данными между различными конфигурациями, необходимо придерживаться следующих рекомендаций по синхронизации классификаторов.

2. Данные классификаторов объектного типа (например, страны мира, ОКЕИ и т.п.) должны сопоставляться по уникальным идентификаторам ссылок, а в случае, если сопоставление по уникальным идентификаторам ссылок не дало положительного результата, то сопоставление должно продолжаться по полям поиска (режим автоматического сопоставления). Для этого требуется:

- в правиле конвертации объекта (ПКО) классификатора установить признаки:
 - "Искать объект приемника по внутреннему идентификатору объекта источника"
 - "Продолжить поиск по полям поиска, если по идентификатору объект приемник не найден"
- указать правила конвертации свойств (ПКС), которые используются в качестве полей поиска. Например, "БИК" для классификатора банков.
- при необходимости определить код обработчика ПКО "Поля поиска" в случае сложного сценария идентификации объектов.

В противном случае, данные классификаторов могут продублироваться, так как у одинаковых с прикладной точки зрения элементов справочника в разных базах, как правило, различные уникальные идентификаторы ссылок.

3. Данные классификаторов, представленные наборами записей (например, адресный классификатор) в обмене участвовать не должны. Данные таких классификаторов должны заполняться и поддерживаться в актуальном состоянии отдельно в каждой информационной базе, между которыми настроен обмен данными. Это позволяет предотвратить избыточную миграцию данных этих классификаторов при обмене данными.

См. также

- [Описание инструмента «Конвертация данных»](#) в разделе «Методическая поддержка 1С:Предприятия 8»

Разработка планов обмена с отборами

Область применения: управляемое приложение, мобильное приложение.

#std701

Рекомендация (полезный совет)

1.1. Как правило, для синхронизации данных между различными конфигурациями и для организации распределенной информационной базы (РИБ) используется технология планов обмена. В этом случае часто возникает задача организации обмена не всеми данными информационной базы, а только частью данных. Например, данные могут быть отобраны для отправки в другие узлы в разрезе организаций, складов, подразделений и пр.

Для определения узлов-получателей для отправки данных следует использовать события [Перед Записью](#) и [Перед Удалением](#) объектов информационной базы, в которых предусмотреть логику регистрации изменений данных на узлах планов обмена (далее – логика регистрации).

При использовании в конфигурации подсистемы «Обмен данными» Библиотеки стандартных подсистем логика регистрации может быть задана декларативно в правилах регистрации объектов (ПРО), которые разрабатываются в конфигурации «Конвертация данных». Подробнее см. документацию к подсистеме «Обмен данными».

1.2. Кроме того, при проектировании логики регистрации необходимо учитывать следующие особенности:

- данные из сообщения обмена загружаются в произвольном порядке, поэтому на момент выполнения логики регистрации необходимые ей данные могут быть еще не загружены;
- логика регистрации в том числе выполняется в контексте загрузки данных в транзакции записи объекта, при этом признак **ОбменДанными.Загрузка** игнорируется. Поэтому любые ошибки логики регистрации приведут к аварийному завершению загрузки данных из сообщения обмена;
- в обмене в распределенной информационной базе (РИБ) могут участвовать не все данные, например, движения в регистрах мигрируют между узлами, а регистраторы (документы) – нет;
- обращение к полям связанных таблиц "через точку" приводят к неявному соединению с дополнительными таблицами, что снижает производительность обмена.

При разработке планов обмена для синхронизации с другими программами (не РИБ, по правилам конвертации) с помощью подсистемы «Обмен данными» Библиотеки стандартных подсистем первые два пункта не требуется учитывать в правилах регистрации для ссылочных типов объектов (СправочникСсылка, ДокументСсылка и пр.), поскольку регистрация ссылочных объектов выполняется отдельной операцией после загрузки всех данных из сообщения обмена.

2. С учетом перечисленных особенностей рекомендуется придерживаться следующих правил.

2.1. Обеспечить самодостаточность данных, участвующих в обмене. Таблицы данных (справочники, документы, регистры и пр.), участвующие в обмене, должны содержать все необходимые данные для выполнения логики регистрации. Логика регистрации не должна обращаться к полям связанных таблиц, она должна оперировать только данными основной таблицы, регистрацию изменений которой необходимо выполнить.

Пример:

- Данные регистрируются в разрезе организаций.
- В обмене участвует две таблицы: документ и регистр накопления, в котором содержатся движения документа. И документ, и регистр накопления являются самодостаточными, т.к. содержит ссылку на организацию.
- При записи данных пользователем или при загрузке данных, логика регистрации отрабатывает независимо для документа и для набора записей регистра.

См. также: [Самодостаточность регистров. Разыменование ссылочных полей составного типа в языке запросов](#)

Если требование самодостаточности данных нельзя поддержать по другим соображениям, то нужно рассмотреть следующие варианты.

2.2. Исключить из обмена вторичные данные. Данные, которые могут быть вычислены независимо в каждой из информационных баз, участвующих в обмене, следует исключить из состава плана обмена. Тем самым, более не требуется логика регистрации этих данных, которая могла обращаться к полям связанных таблиц, участвующих в обмене.

Пример:

- Данные регистрируются в разрезе организаций.
- В обмене участвует только одна таблица – документы поступления товаров.
- Есть регистр сведений, который используется для определения прав доступа пользователей к документам поступления товаров. Этот регистр не является самодостаточным, т.к. не содержит ссылку на организацию, поэтому он не участвует в обмене данными.
- Данные регистра вычисляются независимо в каждом узле распределенной информационной базы при загрузке документов поступления товаров.

2.3. Регистрировать изменения связанных данных. Если в обмене участвуют несколько логически взаимосвязанных таблиц данных, то логика регистрации должна регистрировать к выгрузке данные всех взаимосвязанных таблиц вне зависимости от того, для какой таблицы данных выполняется логика регистрации. При этом логика регистрации не должна аварийно прерываться в случае, когда необходимые для логики регистрации данные отсутствуют.

Пример:

- Данные регистрируются в разрезе организаций.
- В обмене участвует две таблицы: справочник "**Основные средства**" и регистр сведений "**Основные средства организаций**". В регистре сведений хранится связь основного средства и организации, которой принадлежит основное средство.
- При записи элемента справочника, и при записи набора записей регистра к выгрузке регистрируются как элемент справочника, так и набор записей регистра.
- Если при загрузке данных из сообщения обмена сначала выполняется загрузка элемента справочника, то он записывается в базу данных, но не регистрируется к выгрузке на другие узлы планов обмена, т.к. логика регистрации проверяет, что набор записей регистра еще не загружен.
- При последующей загрузке набора записей регистра выполняется регистрация изменений как самого набора записей, так и соответствующего элемента справочника.

Интеграция прикладных решений через формат EnterpriseData

Область применения: управляемое приложение.

#std771

1. Переходы с конфигурации на конфигурацию следует разрабатывать на основе правил конвертации (ПК). Обмены между конфигурациями, использующими **Библиотеку стандартных подсистем** (БСП), следует делать на основе формата **EnterpriseData**, разрабатывать новые обмены на основе ПК запрещается.

2. Актуальные версии **EnterpriseData** содержатся в последней опубликованной версии конфигурации БСП в виде объектов метаданных **Пакет XDTO**, именуемых следующим образом: EnterpriseData_{X|XX}_{Y|YY}_{Z|ZZ}, где X, Y - это Major версия, Z - это Minor версия.

Список актуальных версий поддерживается в состоянии, обеспечивающем оптимальный баланс между затратами на их поддержку в обменах и степенью совместимости версий прикладных решений.

Например, версии 1.0, 1.1 **EnterpriseData** были сняты с поддержки, т.к. интервала 1.2-1.4 версий формата достаточно для обмена между версиями прикладных решений из достаточно широкого диапазона.

3. При разработке новых версий прикладных решений нужно стремиться к тому, чтобы в обменах данными поддерживались все актуальные версии **EnterpriseData**. Это требование обеспечивает возможность асинхронного выпуска новых версий прикладных решений для разработчиков, а также возможность асинхронного перехода на них для пользователей. Исключение составляет случай, описанный в пункте 4.

Запрещается выпускать версию конфигурации, не поддерживающую какую-либо из версий формата, поддерживаемых в версии БСП, встроенной в версию конфигурации.

Пример:

При выпуске конфигурации, в которую встроена версия 2.4.1 БСП, необходимо поддерживать версии 1.2, 1.3 и 1.4 формата **EnterpriseData**.

4. Версия формата **EnterpriseData** не должна поддерживаться в обмене данными, если она не соответствует требованиям к функциональности этого обмена.

Пример:

Для интеграции двух конфигураций необходим обмен документами «Чек ККМ». Поддержка передачи этих данных есть только в версии 1.4 формата, следовательно поддерживаемые в этом обмене версии формата должны быть не младше 1.4.

Методическая рекомендация (полезный совет)

5. В редких случаях обмен данными между прикладными решениями невозможен по причине отсутствия совместно поддерживаемых версий **EnterpriseData**. Например, такая ситуация может возникнуть, когда версия одного из прикладных решений, сильно устарела и требует обновления. В целях предотвращения данной проблемы рекомендуется информировать пользователей о том, какие версии **EnterpriseData** поддерживаются в прикладном решении.

См. также

- [Обмен данными с прикладными решениями на платформе “1С:Предприятие” в формате EnterpriseData](#) в разделе «Разработка и администрирование» ИТС

Разработка конфигураций с повторным использованием общего кода и объектов метаданных

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

#std551

Методическая рекомендация (полезный совет)

1. В условиях разработки нескольких конфигураций (например, линейки продуктов) возникает задача повторного использования общего кода и объектов конфигурации. Этую задачу рекомендуется решать с использованием вспомогательных конфигураций – библиотек.

Библиотечная конфигурация (библиотека) – это конфигурация, которая в отличие от «обычных» прикладных решений не предназначена для использования конечными пользователями, а служит только для поддержки конфигураций, пользующихся ее функциональностью. В них размещается функциональность, общая для прикладных решений.

Библиотечный подход к разработке общей функциональности прикладных решений позволяет:

- вести разработку общей функциональности централизованно (а не в каждом прикладном решении «по месту»);
- выпускать версии общей функциональности в виде продукта (дистрибутива или файла поставки библиотеки);

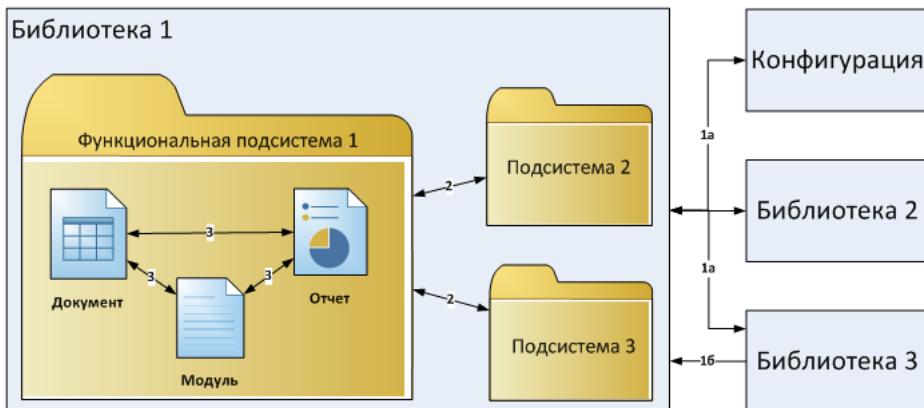
- повторно использовать код и объекты библиотеки в конфигурациях с помощью механизма постановки на поддержку платформы **1С:Предприятия**;
- унифицировать конфигурации по набору общих объектов метаданных и функциональных подсистем.

2. В свою очередь, библиотека может стоять на поддержке у другой библиотеки, образуя таким образом **иерархию библиотек**. Библиотечная конфигурация, которая стоит на поддержке у другой библиотеки (**библиотеки-родителя**), называется **библиотекой-наследником**.

Как правило, библиотеки не являются самостоятельным предметом разработки, а предназначены только для создания прикладных решений. Таким образом, прикладные решения являются конечными конфигурациями-потребителями в иерархии библиотек.

3. При взаимодействии конфигураций, библиотек и внешних потребителей следует выделять следующие области видимости программного кода:

- Программный интерфейс** содержит экспортные процедуры и функции, предназначенные для использования сторонними потребителями. Программный интерфейс можно разделить на две категории:
 - Программный интерфейс для использования любыми внешними потребителями предназначен для вызова из произвольного места в коде конфигурации. Потребителями такого программного интерфейса чаще всего выступают расширения конфигурации, сторонние доработки конфигурации или другие программы.
 - Программный интерфейс для конкретных потребителей должен располагаться в определенном месте в коде конфигурации, определенном в документации, и может быть вызван только определенным потребителем. Такой программный интерфейс рекомендуется размещать в отдельном подразделе **Для вызова из других подсистем**. Подробнее см. стандарт [Обеспечение обратной совместимости библиотек](#).
- Служебный программный интерфейс** предназначен для экспортных процедур и функций, которые допустимо вызывать только из других функциональных подсистем этой же библиотеки (конфигурации).
- Служебные процедуры и функции** содержат внутреннюю реализацию функциональной подсистемы. Экспортные процедуры и функции, расположенные в этом разделе, предназначены только для вызова из других объектов этой же подсистемы.



См. также

- [Использование дублирующего кода](#)
- [Переопределяемые и поставляемые объекты библиотеки](#)
- [Обеспечение обратной совместимости библиотек](#)

Имена объектов метаданных в иерархии библиотек

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

#std552

1. При разработке конфигурации в условиях использования одной или нескольких **библиотек** следует избегать конфликтов имен объектов метаданных (справочников, документов, общих модулей и др.), поставляемых из библиотек, с объектами самой конфигурации.

Для разрешения таких конфликтов за библиотекой более «низкого» уровня рекомендуется организационно закрепить преимущественное право выбора названия. Если в одном из ее потребителей оказывается объект с таким же именем, то следует переименовать объект в потребителе с использованием более конкретного названия.

Например: если в конфигурации «Управление предприятием» имеется группа команд **Настройки**, конфликтующая с одноименной группой команд из библиотеки «Базовая функциональность», то для группы команд в конфигурации-потребителе следует выбрать более точное название **НастройкиУправленческогоУчета** или **НастройкиРасчетаСебестоимости** или что-то иное.

2. При разработке независимых друг от друга библиотек (одноуровневых) такие конфликты могут быть выявлены достаточно поздно – при или после их внедрения в конфигурацию-потребитель.

Для того чтобы снизить вероятность возникновения конфликтов в этом случае, следует всегда предметно называть объекты метаданных. Название должно кратко описывать сущность объекта; не должно называться слишком «универсально».

2.1. Наиболее «универсальные» названия объектов метаданных следует использовать в библиотеках самого нижнего уровня иерархии. Например, общие модули **ОбщегоНазначения**, **СтроковыеФункции**, **РаботаСФормами** и т.п.

2.2. В конечных конфигурациях-потребителях и в библиотеках более высокого уровня использовать более специфичные для данной прикладной области названия. Например, общие модули **ТорговыйУчет**, **ЗарплатаКадры**, **ПроведениеСкладскихДокументов** и т.п.

2.3. Функционал, который расширяет аналогичный функционал библиотеки более низкого уровня или конфликтует с «соседней» одноуровневой библиотекой, рекомендуется именовать с добавлением уточняющих постфиксов, которые соответствуют имени конкретной библиотеки или конечной конфигурации-потребителя. Например:

- общий модуль **ОбновлениеИнформационнойБазыМФ** – функции обновления ИБ, специфичные для прикладного решения, конечной конфигурации-потребителя «Мои финансы» (постфикс **МФ**)
- общий модуль **СтроковыеФункцииРегл** – функции общего назначения, специфичные для библиотеки регламентированного учета (постфикс **Регл**)
- общий модуль **РаботаСФормамиЗарплатаКадры** – функции для работы с формами в библиотеке базовой зарплатно-кадровой функциональности (постфикс **ЗарплатаКадры**)
- подписка на событие **УстановитьПометкуУдаленияПрисоединенныхФайловЗарплатаКадры** - определена в библиотеке базовой зарплатно-кадровой функциональности.

Переопределяемые и поставляемые объекты библиотеки

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

#std553

1. С точки зрения возможностей по настройке функциональности **библиотеки** для работы в конкретной конфигурации-потребителе все объекты библиотеки условно делятся на три категории:

- Непереопределяемые объекты** – «готовые» объекты, которые можно брать и использовать в конфигурации-потребителе «как есть». Их не следует изменять в конфигурации-потребителе, чтобы они были гарантированно одинаковы во всех конфигурациях, основывающихся на данной библиотеке. Более того, такие объекты обязательно должны присутствовать в конфигурациях, использующих библиотеку. Примеры: справочник **Пользователи**.

- **Переопределяемые объекты** – «изменяемые» объекты для настройки библиотеки под конкретную конфигурацию. Они могут или должны быть изменены в конфигурации-потребителе. С помощью таких объектов решаются задачи изменения поведения библиотечной функциональности, ее параметризации спецификой конфигурации-потребителя, а также для подключения библиотечной функциональности к объектам конфигурации-потребителя.
- **Определители типов** – объекты-«классификаторы», которые не имеют базовой реализации. Предназначены для формирования единого пространства имен в конфигурациях, а реализация при этом может как угодно сильно различаться в конфигурациях-потребителях. Например: справочники-классификаторы, в которых определено только «название»; сущность «корганизация» должна быть везде представлена справочником с именем **Организация** и т.п.

2. Рекомендуется устанавливать для объектов этих категорий следующие **правила поставки**:

- Непереопределяемые объекты – «изменения не рекомендуются»;
- Переопределяемые объекты и определители типов – «изменения разрешены».

Эти рекомендации продиктованы следующими соображениями:

- Непереопределяемые объекты – это зона ответственности разработчиков библиотеки, поэтому они не должны разрабатываться «по месту» в конфигурациях-потребителях. Но если необходимость изменений носит срочный характер (например, исправление критичной ошибки) или продиктована какими-то другими особыми соображениями, то допускается вносить изменения в непереопределяемые библиотечные объекты непосредственно в конфигурациях-потребителях. При этом нужно иметь в виду, что эти изменения могут быть потеряны при следующем обновлении версии библиотеки в конфигурации-потребителе, если не принять специальные меры (донести до разработчиков библиотеки необходимость внесения изменений или задокументировать этот отход от общей инструкции по обновлению библиотеки).
- Переопределяемые объекты и определители типов должны или могут быть изменены в конфигурации-потребителе, исходя из их назначения.

3. Для того чтобы упростить настройку библиотеки и снизить трудоемкость последующих обновлений версии библиотеки в конфигурации-потребителе следует **минимизировать количество переопределяемых объектов** с помощью следующих методик:

- Настройка состава типов переопределяемых реквизитов (свойств) тех или иных объектов библиотеки – для подключения библиотечной функциональности к объектам конфигурации-потребителя.
Например: можно подключить библиотечную функциональность к конкретным объектам конфигурации с помощью расширения состава типов общей команды, измерения составного регистра сведений и т.п.
- Добавление предопределенных элементов – для параметризации библиотечной функциональности спецификой конфигурации-потребителя.
Например: для библиотечной подсистемы ведения и обработки контактной информации с помощью предопределенных элементов библиотечного справочника **ВидыКонтактнойИнформации** можно указать, какие виды контактной информации (телефон, адрес, электронный адрес и т.п.) должны быть предусмотрены для объектов конфигурации-потребителя.
- Переопределяемые общие модули – для изменения поведения библиотечной функциональности в конкретной конфигурации-потребителе
 - с помощью переопределения тех или иных «обработчиков событий», предоставляемых библиотекой; например:
ПриПодготовкеМакетаОписанияОбновлений
ПриЗаписиСпискаБизнесПроцессов
 - а также для того, чтобы сообщить ту или иную информацию из конфигурации-потребителя в библиотеку. Например:
ПриОпределенииБазовойВерсииКонфигурации
ПриДобавленииОбработчиковОбновления

3.1. Переопределяемые общие модули следует называть с постфиксом **Переопределяемый**.

См. также: [правила создания общих модулей](#)

3.2. Переопределяемые общие модули должны содержать только экспортные процедуры, которые вызываются из кода самой библиотеки. Другими словами, не следует допускать вызовов процедур переопределяемых модулей непосредственно из кода конфигурации-потребителя.

Такое ограничение обусловлено соображением повышения устойчивости кода конфигурации, который вызывает библиотечные процедуры и функции, составляющие программный интерфейс библиотеки. К программному интерфейсу библиотеки следует относить только экспортные процедуры и функции непереопределяемых общих модулей.

Например, в библиотеке имеются модули **ПапкиФайлов** и **ПапкиФайловПереопределяемый**. Для использования в конфигурациях-потребителях в модуле **ПапкиФайлов** реализуется экспортная функция:

Функция ПапкаФайлов(ВладелецФайловСсылка) Экспорт

```
СтандартнаяОбработка = Истина;
Результат = Неопределен;
ПапкиФайловПереопределяемый.ПриПолученииПапкиФайлов(ВладелецФайловСсылка, Результат, СтандартнаяОбработка);

Если СтандартнаяОбработка Тогда
  // реализация по умолчанию
  Результат = ...
КонецЕсли;
Возврат Результат;
```

КонецФункции

а в модуле **ПапкиФайловПереопределяемый** - процедура-обработчик **ПриПолученииПапкиФайлов**:

```
// Вызывается из библиотеки при необходимости получить папку файлов для указанного владельца.
// Параметры:
// ВладелецФайловСсылка - ЛюбаяСсылка           - владелец файлов, для которого нужно вернуть папку.
// ПапкаФайлов          - СправочникСсылка.ПапкиФайлов - в этот параметр нужно записать результат.
// СтандартнаяОбработка - Булево                   - по умолчанию, Истина. В этом случае папка будет получена способом по умолчанию.
// Если значение параметра установить в Ложь, то в этой процедуре можно реализовать свой способ,
// которым в конфигурации получают папки файлов.
//
Процедура ПриПолученииПапкиФайлов(ВладелецФайловСсылка, ПапкаФайлов, СтандартнаяОбработка) Экспорт
```

КонецПроцедуры

При этом все вызовы из конфигурации-потребителя должны идти только к библиотечному модулю **ПапкиФайлов**. Обращение к **ПапкиФайловПереопределяемый** разрешается только из библиотечного модуля **ПапкиФайлов**.

3.3. При этом в переопределяемом модуле следует располагать только экспортные процедуры с пустой реализацией. В нем не должно быть каких-либо других не-экспортных процедур или функций. Базовую реализацию переопределяемых процедур и функций следует располагать в непереопределяемом коде.

Такое ограничение вызвано необходимостью снизить трудоемкость последующих обновлений переопределяемых модулей в конфигурации-потребителе. Например, неправильно поставлять переопределяемый модуль **МояБиблиотекаПереопределяемый** с какой-либо реализацией:

Функция НастройкаПараметровРаботы() Экспорт

```
ПараметрыРаботы = Новый Структура;
// если настройки по умолчанию не подходят, то измените их.
ПараметрыРаботы.Вставить("ПоказыватьЕдинственныйРаздел", Ложь);
ПараметрыРаботы.Вставить("ЗадаватьДатуДляПрочихРазделов", Ложь);
ПараметрыРаботы.Вставить("ИспользоватьВнешнихПользователей", Ложь);
Возврат ПараметрыРаботы;
```

КонецФункции

правильно:

```
// Позволяет настроить работу подсистемы.
//
```

```
// Параметры:  
// ПараметрыРаботы - Структура - содержит свойства:  
//   * ПоказыватьЕдинственныйРаздел - Булево - по умолчанию Ложь.  
//   * ЗадаватьдатудляПрочихРазделов - Булево - по умолчанию Ложь.  
//   * ИспользоватьВнешнихПользователей - Булево - по умолчанию Ложь.  
//  
Процедура ПриПолученииНастроекПараметровРаботы(ПараметрыРаботы) Экспорт
```

КонецПроцедуры

а установку значений по умолчанию перенести в непреопределляемый общий модуль библиотеки:

Функция НастройкаПараметровРаботы()

```
ПараметрыРаботы = Новый Структура;  
// настройки по умолчанию  
ПараметрыРаботы.Вставить("ПоказыватьЕдинственныйРаздел", Ложь);  
ПараметрыРаботы.Вставить("ЗадаватьдатудляПрочихРазделов", Ложь);  
ПараметрыРаботы.Вставить("ИспользоватьВнешнихПользователей", Ложь);  
  
// а теперь запросим конфигурацию-потребитель на случай,  
// если эти умолчания не устраивают  
МояБиблиотекаПреопределляемый.ПриПолученииНастроекПараметровРаботы(ПараметрыРаботы);  
Возврат ПараметрыРаботы;
```

КонецФункции

3.4. При обновлении версии библиотеки в конфигурации-потребителю особого внимания требуют модули корневого объекта конфигурации и переопределываемые общие модули, так как автоматическое обновление таких «узких мест» конфигурации-потребителя невозможно. Для настройки в конфигурации переопределываемых общих модулей рекомендуется придерживаться общего подхода:

- При первой настройке переопределяемого общего модуля следует ознакомиться с документацией к его экспортным процедурам и функциям, приведенной в комментариях к ним. И при необходимости вписать реализацию в экспортные процедуры и функции модуля.
- При каждом обновлении переопределяемого общего модуля требуется перенести новые экспортные процедуры и функции, удалить устаревшие и убедиться, что комментарии, количество и имена параметров у всех функций совпадают с их библиотечными эквивалентами. При необходимости вписать реализацию в новые экспортные процедуры и функции модуля, а также актуализировать реализацию уже существующих функций, если в новой версии библиотеки было изменено их назначение или состав параметров.

См. также

- [Переопределение общих модулей в условиях иерархии библиотек](#)

Отнесение объектов библиотек к подсистемам

Область применения: управляемое приложение, обычное приложение.

#std705

1. Все объекты библиотеки должны быть отнесены к одной корневой подсистеме (например, для объектов БСП – это подсистема **СтандартныеПодсистемы**), которая не содержит командного интерфейса.

2. Допустимо наличие нескольких корневых подсистем (помимо упомянутой в п.1), содержащих командный интерфейс.

См. также

- [Использование подсистем](#)

Переопределение общих модулей в условиях иерархии библиотек

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

#std554

Методическая рекомендация (полезный совет)

При разработке нескольких библиотек, стоящих на поддержке друг у друга, следует минимизировать трудоемкость по обновлению кода [переопределяемых общих модулей](#) в каждой из библиотек. Для этого рекомендуется использовать следующую методику:

1. В библиотеке нижнего уровня иерархии (самой базовой) объявляется переопределяемый модуль по правилам [создания общих модулей](#).
Например: **БазоваяБиблиотекаПреопределляемый**.

2. В библиотеках более высокого уровня реализация процедур и функций переопределяемого модуля базовой библиотеки может быть при необходимости доопределена. При этом в переопределяемом модуле размещается не сама реализация (ее код), а только один вызов.

Например, в общем модуле **БазоваяБиблиотекаПреопределляемый** в библиотеке верхнего уровня «Базовая библиотека» реализация фактически размещается в модуле **МодульБазовойБиблиотеки**:

Процедура НастроитьИнтерфейс(Знач ПараметрыРаботы) Экспорт

```
// Начало: Базовая библиотека  
МодульБазовойБиблиотеки.НастроитьИнтерфейс(ПараметрыРаботы);  
// Конец: Базовая библиотека
```

КонецПроцедуры

в этом же общем модуле **БазоваяБиблиотекаПреопределляемый** в следующей библиотеке верхнего уровня «Библиотека второго уровня»:

Процедура НастроитьИнтерфейс(Знач ПараметрыРаботы) Экспорт

```
// Начало: Базовая библиотека  
МодульБазовойБиблиотеки.НастроитьИнтерфейс(ПараметрыРаботы);  
// Конец: Базовая библиотека  
  
// Начало: Библиотека второго уровня  
МодульБиблиотекиВторогоУровня.НастроитьИнтерфейс(ПараметрыРаботы);  
// Конец: Библиотека второго уровня
```

КонецПроцедуры

и т.д.

3. Таким образом, в конечной конфигурации-потребителю реализация переопределяемого модуля имеет вид:

Процедура НастроитьИнтерфейс(Знач ПараметрыРаботы) Экспорт

```

// Начало: Базовая библиотека
МодульБазовойБиблиотеки.НастройИнтерфейс(ПараметрыРаботы);
// Конец: Базовая библиотека

// Начало: Библиотека второго уровня
МодульБиблиотекиВторогоУровня.НастройИнтерфейс(ПараметрыРаботы);
// Конец: Библиотека второго уровня

// а теперь выполняем настройки так, как это надо нашей конфигурации
// ...

```

КонецПроцедуры

Такой подход позволяет

- скрыть от библиотек верхнего уровня и конфигураций-потребителей детали реализации библиотек более нижнего уровня, что минимизирует риск ошибки при обновлении кода переопределяемого общего модуля;
- но при этом, в общем случае, оставляет возможность выбора: воспользоваться или отказаться от нее.

Размещение сведений о настройках подсистемы

Область применения: управляемое приложение, обычное приложение.

#std739

Для задействования возможностей по настройке функциональности [библиотеки](#) для работы в конкретной конфигурации-потребителе используются [переопределяемые объекты](#).

Все настройки подсистемы, доступные для изменения разработчикам на внедрении, можно разделить на общие для всех объектов подсистемы (или для группы объектов) и специфичные для конкретных объектов. При выборе места размещения таких настроек следует руководствоваться следующими принципами.

1. Настройки подсистемы, общие для всех объектов или для группы объектов метаданных, рекомендуется задавать в переопределяемом модуле. Для этого создать одну процедуру с именем **ПриОпределенииНастроек** и параметром **Настройки** типа **Структура**, которая предварительно заполнена значениями по умолчанию. Использования структуры в качестве значения параметра позволяет расширять программный интерфейс, добавляя новые свойства в структуру, без нарушения [обратной совместимости](#).

Например, определение настроек подсистемы Варианты отчетов, общих для всех отчетов в общем модуле ВариантыОтчетовПереопределляемый:

```

Процедура ПриОпределенииНастроек(Настройки) Экспорт
Настройки.ВыводитьОтчетыВместеВариантов = Истина;
Настройки.ВыводитьОписания = Истина;
Настройки.ДругиеОтчеты.ПоказыватьФлажок = Истина;
КонецПроцедуры

```

1.1. Для определения списка объектов конфигурации, с которыми работает подсистема, не следует использовать перебор объектов метаданных с [конструкцией Попытка Исключение](#). Вместо этого, рекомендуется создавать отдельную процедуру переопределяемого модуля, в которой перечислять такие объекты. Например, **ПриОпределенииПодключенныхОтчетов** содержит в себе список отчетов конфигурации, подключенных к подсистеме.

1.2. Процедуры, позволяющие переопределить бизнес-логику сразу для всех объектов, с которыми работает подсистема, также рекомендуется размещать в переопределяемом модуле. Имена процедур рекомендуется выбирать исходя из выполняемого действия. Например: **ПриРегистрацииИзмененийКлючейВариантовОтчетов** описывает изменения имен вариантов отчетов. Для каждого выполняемого действия рекомендуется создавать отдельную процедуру.

2. Настройки или обработчики, специфичные для отдельного объекта, рекомендуется размещать в модуле менеджера этого объекта. Для процедуры, содержащей в себе информацию о свойствах, реквизитах или о наличии методов объекта рекомендуется использовать имена вида **ПриОпределенииНастроек<ИмяПодсистемы>**. Например **ПриОпределенииНастроекВариантовОтчетов**.

Процедуры, создающие переопределяемую бизнес-логику, специфичную для конкретного объекта так же рекомендуется размещать в модуле менеджера этого объекта. Для каждого выполняемого действия рекомендуется создавать отдельную процедуру. При этом наличие той или иной процедуры в модуле менеджера задается в процедуре **ПриОпределенииНастроек<ИмяПодсистемы>**, что позволяет избежать использования [конструкции Попытка Исключение](#) для проверки наличия процедуры. Например, в модуле менеджера отчета заданы процедуры:

```

Процедура ПриОпределенииНастроекВариантовОтчетов(Настройки) Экспорт
Настройки.ОпределитьНастройкиФормы = Истина;
Настройки.Размещение.Вставить(ВариантыОтчетовКлиентСервер.ИдентификаторНачальнойСтраницы(), "Важный");
Настройки.ПолучениеДанныхОтчета = Истина; // Наличие процедуры.
КонецПроцедуры

```

```

Процедура ПриПолучениеДанныхОтчета(данные) Экспорт
данные.Очистить();
// Своя процедура получения данных.
КонецПроцедуры

```

Процедура **ПриОпределенииНастроекВариантовОтчетов** определяет настройки конкретного отчета и сообщает о наличии процедуры **ПриПолучениеДанныхОтчета**, в которой задана переопределяемая обработка данных отчета.

Обеспечение совместимости библиотек

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

#std644

1. При разработке библиотек необходимо обеспечивать обратную совместимость (далее просто: совместимость) между различными версиями библиотек в пределах одной [подредакции](#) библиотеки.

Например, версии библиотеки 2.0.1, 2.0.2 и 2.0.5 должны быть совместимы. Однако допустимо, если следующая подредакция 2.1 будет содержать существенные изменения, нарушающие это правило.

Совместимость версий библиотек позволяет существенно минимизировать затраты на обновление библиотеки в конфигурациях-потребителях, так как не требует от прикладных разработчиков многократно пересматривать код и адаптировать объекты метаданных своих конфигураций под изменения библиотеки. Прикладное решение может «уверенно» использовать старые возможности библиотеки, не «торопясь» переходить на новые.

Кроме того, при разработке нескольких библиотек, стоящих на поддержке друг у друга, совместимость позволяет вести совместную разработку «соседних» библиотек на разных версиях базовой библиотеки, без необходимости частого обновления всего дерева библиотек.

1.1. Таким образом, полный номер версии библиотеки однозначно указывает на характер изменений и совместимость с ее предыдущими версиями:

Что изменилось в новой версии?	РР	.ПП	.ВВ	.СС
Архитектурные изменения, нарушили совместимость (есть инструкция по переходу)	✓	✓	✗	✗
Новые функции	✓	✓	✓	✗
Исправлены ошибки	✓	✓	✓	✓

Здесь:

1. РР (редакция) – существенно нарушена совместимость (серьезные архитектурные или «знаковые» изменения в библиотеке);
2. ПП (номер подредакции) – нарушена совместимость (требуется отработать инструкцию по переходу на эту версию, иначе конфигурация будет неработоспособна);
3. ВВ (номер версии) – доступны только новые функции для пользователей и/или разработчиков (возможна «механическая» автозаливка метаданных библиотеки в конфигурацию-потребитель; инструкция по переходу на эту версию не обязательна, без ее отработки конфигурация сохранит работоспособность);
4. СС (номер сборки) – содержит только исправление ошибок (возможна «механическая» автозаливка метаданных библиотеки в конфигурацию-потребитель).

Исключением являются заранее согласованные с потребителями библиотеки случаи, такие как отработка изменений законодательства. При отработке изменений законодательства (вступающего в силу до публикации следующей версии) следует выпускать исправительные сборки с этими изменениями для всех веток, находящихся на поддержке, даже если эти изменения добавляют новые функции или нарушают совместимость. В таких случаях вынужденное нарушение обратной совместимости должно быть отражено в сопроводительной документации согласно п.1.8, отработка которой обязательна.

1.2. В библиотеках с несколькими [функциональными подсистемами](#) нарушение совместимости хотя бы в одной подсистеме означает увеличение номера подредакции библиотеки (2-я цифра).

1.3. При незапланированном нарушении совместимости в 3-й и 4-й цифре следует отозвать релиз библиотеки, устраниТЬ нарушение и выпустить новый исправленный релиз.

1.4. Аналогичные требования распространяются и на незапланированное добавление новых функций в 4-й цифре.

1.5. В целях обеспечения совместимости следует

- выделить программный интерфейс библиотеки, скрыв от потребителей все остальные детали ее реализации;
- и не изменять программный интерфейс и поведение. Его допустимо только расширять.

Требования обеспечения обратной совместимости имеют приоритет над следующими стандартами разработки:

- [Имя, синоним комментарий](#)
- [Названия процедур и функций](#)

1.6. К программному интерфейсу библиотеки относятся те ее объекты метаданных, которые предназначены для использования в прикладном коде:

- имена, состав параметров и поведение экспортных процедур и функций, которые размещены в [разделе «программный интерфейс»](#);
- имена, состав параметров и поведение всех экспортных процедур [переопределляемых общих модулей](#);
- имена объектов метаданных (включая их реквизиты, табличные части и пр.), к которым допускается непосредственное обращение из прикладного кода или из запросов.

1.7. Совместимость не нарушает расширение программного интерфейса библиотеки, т.е. в 3-й цифре допустимо, например:

- добавление новой функции в программный модуль;
- добавление процедуры в переопределяемый модуль;
- добавление еще одного необязательного параметра в конец списка формальных параметров существующей функции;
- добавление нового реквизита в справочник, регистр и т.п. (к которым библиотека допускает прямые обращения).

В то же время:

- Добавления нового API и расширение состава параметров существующего API не ожидают от исправительных релизов библиотеки (поэтому недопустимо в 4-й цифре);
- Даже если состав параметров и название процедуры не меняется, но изменилось ее поведение, то это является нарушение совместимости (т.е. допустимо только во 2-й цифре)
 - например, процедура **Сумма(а, б)** вместо суммы вдруг стала вычислять разность.

1.8. В остальных случаях, когда согласно п.1 допустимо отказаться от поддержки совместимости, следует документировать в сопроводительной документации к библиотеке любые изменения, приводящие к нарушению совместимости. Документация должна включать инструкцию для прикладных разработчиков по адаптации своих конфигураций к новому программному интерфейсу библиотеки.

Примеры фрагментов документации:

- Общий модуль **ВнешниеЗадачиПереопределяемыйВызовСервера** переименован в **ВнешниеЗадачиВызовСервераПереопределяемый**. Необходимо заменить все обращения к этому модулю в коде конфигурации.
- Процедура **УстановитьПроизвольныйЗаголовокПриложения** общего модуля **СтандартныеПодсистемыКлиент** переименована в **УстановитьРасширенныйЗаголовокПриложения**. Необходимо заменить все обращения к этой процедуре в коде конфигурации.
- Отчет **СправкаПоИсполнительскойДисциплине** удален. Вместо него следует использовать одноименный вариант отчета **Задачи**. Необходимо заменить все обращения к этому отчету в коде и в метаданных конфигурации.
- В общий модуль **ЗащитаПерсональныхДанныхПереопределяемый** добавить процедуру **ДополнитьДанныеОрганизацииОператораПерсональныхДанных**, перенеся ее определение из поставки библиотеки.
- Хранение предмета взаимодействий перенесено из реквизита документа в реквизит **Предмет** регистра сведений **ПредметыПапкиВзаимодействий**. Необходимо заменить все обращения к реквизиту **Предмет** документов взаимодействий на реквизит **Предмет** регистра сведений.

1.9. Рекомендуется размещать программный интерфейс библиотеки только в ее общих модулях, а не в модулях объектов, менеджеров, наборов записей и т.п.

2.1. Для разделения программного интерфейса от служебных процедур и функций необходимо размещать их в разных разделах модуля или в разных общих модулях.

При размещении в разных общих модулях, к модулям со служебными процедурами и функциями может быть добавлен постфикс **Служебный** (англ. **Internal**).

Например:

- Общие модули **ОбменСообщениями** и **ОбменСообщениямиКлиент** – программный интерфейс подсистемы «Обмен сообщениями»
- Общий модуль **ОбменСообщениямиСлужебный** – служебные процедуры и функции подсистемы, которые не предназначены для использования в коде конфигурации-потребителя.

Такое размещение программного интерфейса в отдельных общих модулях позволяет

- сосредоточить весь программный интерфейс библиотеки в относительно небольшом, обозримом количестве общих модулей,
- а также выводить в контекстной подсказке при вводе текстов модулей только те процедуры и функции, которые действительно являются частью программного интерфейса. Например, после точки в строке «**ОбменСообщениями.**» в редакторе текста модуля разработчик конфигурации-потребителя увидит только то, что ему действительно может понадобиться в работе.

2.2. Раздел **Программный интерфейс** так же может содержать в себе процедуры и функции, предназначенные для вызова конкретными потребителями из других функциональных подсистем библиотеки или из других библиотек. Такие процедуры и функции рекомендуется выделять в отдельный подраздел **Для вызова из других подсистем**, оформленный в виде области **ДляВызоваИзДругихПодсистем** (англ. **InterfaceImplementation**).

Внутри подраздела процедуры и функции должны быть разделены на группы комментариями с именем потребителя, для которого они предназначены. Подробнее см. [Разработка конфигураций с повторным использованием общего кода и объектов метаданных](#). Например:

#Область ПрограммныйИнтерфейс
//Код процедур и функций

#Область ДляВызоваИзДругихПодсистем

// СтандартныеПодсистемы.ГрупповоеИзменениеОбъектов
// Код процедур и функций
// Конец СтандартныеПодсистемы.ГрупповоеИзменениеОбъектов

// ТехнологияСервиса.ВыгрузкаЗагрузкаДанных
// Код процедур и функций
// Конец ТехнологияСервиса.ВыгрузкаЗагрузкаДанных

#КонецОбласти

#КонецОбласти

Англоязычный вариант синтаксиса:

```
#Region Public
// Enter code here.

#Region InterfaceImplementation
// StandardSubsystems.BatchObjectModification
// Enter code here.
// End StandardSubsystems.BatchObjectModification

// SaaSTechnology.DataExportImport
// Enter code here.
// End SaaSTechnology.DataExportImport

#EndRegion
#EndRegion
```

3. Для обеспечения совместимости программного интерфейса библиотеки в условиях активного развития ее функциональности ниже приведен ряд практических рекомендаций.

3.1. При необходимости переименовать (или удалить) экспортную функцию (процедуру) следует оставить прежнюю реализацию функции, пометив ее как устаревшую с помощью комментария вида:

```
// Устарела: Следует использовать функцию ПересчитатьПоКурсу
// ...
Функция ПересчитатьИзВалютыВВалюту(Сумма, ВалютаНач, ВалютаКон, ПоКурсуНач, ПоКурсуКон, ПоКратностьНач = 1, ПоКратностьКон = 1)
Экспорт
```

(англоязычный аналог "Устарела" в начале комментария - "Deprecated")

и разместить новую версию функции с новым именем (в данном примере - **ПересчитатьПоКурсу**).

При этом устаревшую функцию следует перенести в **область общего модуля УстаревшиеПроцедурыИФункции** (англ. **Deprecated**), которая размещена внутри области **ПрограммныйИнтерфейс**. В процедурах и функциях, размещенных в области **УстаревшиеПроцедурыИФункции**, допустимы отклонения от других стандартов разработки согласно п.1.1.

В этом случае, существующий прикладной код не потребуется переписывать. При этом, если при выпуске новой редакции библиотеки будет принято решение удалить все устаревшие функции, то такие функции могут быть легко выявлены в коде библиотеки и удалены.

По каждой устаревшей функции в сопроводительной документации к библиотеке также даются рекомендации по их замене следующего вида:

- Процедура **ПересчитатьИзВалютыВВалюту** общего модуля **ОбщегоНазначения** устарела, вместо нее следует использовать **ПересчитатьПоКурсу**. Устаревшая функция оставлена для обратной совместимости.

3.2. В ряде случаев даже при исправлении ошибки, которое поменяло поведение экспортной функции (процедуры), рекомендуется оставлять прежнюю ошибочную функцию, пометив ее как устаревшую, и размещать исправленную версию функции с новым именем. Это позволяет обеспечить работоспособность того прикладного кода, который ранее заложился на неправильное поведение, но работает корректно. При этом его немедленная переработка нецелесообразна или вообще нежелательна (например, в силу большого количества мест вызовов).

Более того, может быть крайне нежелательным изменение поведения даже в ненштатных и незадокументированных случаях вызова экспортной функции (процедуры).

Например, вызывающий код при некорректных значениях входных параметров функции API ожидает (обрабатывает) незадокументированное возвращаемое значение Неопределено, а в новой версии библиотеки эта функция была исправлена и стала вызывать исключение в этом случае. Несмотря на то, что новое поведение спроектировано как более корректное и даже стало задокументированным, это может привести к массовым ошибкам во всех местах ее вызова.

3.3. При необходимости пересмотреть состав параметров экспортных функций (процедур) следует использовать опциональные параметры, которые добавляются в конец списка формальных параметров.

Например:

```
Функция ПересчитатьИзВалютыВВалюту(Сумма, ВалютаНач, ВалютаКон, ПоКурсуНач, ПоКурсуКон, ПоКратностьНач = 1, ПоКратностьКон = 1)
Экспорт
```

При этом в случае большого числа параметров рекомендуется предусмотреть последний параметр типа **Структура**, состав свойств которой можно безболезненно расширять в дальнейшем:

```
Функция ПересчитатьИзВалютыВВалюту(Сумма, ВалютаНач, ВалютаКон, ПоКурсуНач, ПоКурсуКон, ПараметрыПересчета = Неопределено) Экспорт
```

Такой подход рекомендуется использовать заблаговременно в тех процедурах и функциях, где с высокой степенью вероятности ожидается увеличение количества параметров и режимов работы.

3.4. Параметры типа **Структура** применимы и для сохранения совместимости программного интерфейса, через который библиотека обращается к объектам конфигурации-потребителя. Например, процедура **ПриОпределенииНастроек** позволяет библиотеке добавлять новые режимы работы без потери совместимости:

```
Процедура ПриОпределенииНастроек(Настройки) Экспорт
Настройки.ВыводитьОписания = Истина;
Настройки.События.ПриСозданииНаСервере = Истина;
Настройки.... = ...;
КонецПроцедуры
```

Кроме того, этот подход позволяет также сохранять совместимость программного интерфейса, через который библиотека обращается к объектам конфигурации-потребителя. В примере выше строка «**Настройки.События.ПриСозданииНаСервере = Истина;**» означает, что в конфигурации-потребителя определен одноименный обработчик события библиотеки, который следует вызывать из библиотеки. При этом появление нового события в следующей версии библиотеки не потребует обязательного добавления его пустых обработчика во всех конфигурациях-потребителей. Аналогичным образом, платформа 1С:Предприятие не требует вставлять пустые «заглушки» стандартных обработчиков событий в модули и менеджеры объектов.

3.5. Для минимизации ситуаций, когда в конфигурациях-потребителях возникает потребность в прямом обращении к объектам метаданным библиотеки (реквизитам, табличным частям справочников, документов и пр.), следует предусмотреть в библиотеке программный интерфейс, посредством которого прикладной код может взаимодействовать с библиотекой. Это снижает зависимость прикладного кода от особенностей реализации библиотеки и, тем самым, повышает его устойчивость к обновлениям на новые версии библиотеки.

Например, вместо «прямого» запроса к библиотечному регистру из прикладного кода:

```
Запрос = Новый Запрос;
Запрос.Текст = "ВЫБРАТЬ
| ОбластиДанных.Представление
| ИЗ
| РегистрСведений.ОбластиДанных КАК ОбластиДанных
| ГДЕ
| ОбластиДанных.ОбластьДанных = &ОбластьДанных";
Запрос.УстановитьПараметр("ОбластьДанных", ПараметрыСеанса.ОбластьДанныхЗначение);
ТаблицаОбластейДанных = Запрос.Выполнить().Выгрузить();
ИмяПриложения = ?(ТаблицаОбластейДанных.Количество() = 0, "", ТаблицаОбластейДанных.Получить(0).Получить(0));
```

следует предусмотреть в библиотеке экспортную функцию, которая специально предназначена для использования в прикладном коде:

```
ИмяПриложения = РаботаВМоделиСервиса.ИмяПриложения();
```

При этом если в текущей версии библиотеки отсутствует специализированная функция, а потребность обращаться к ее данным есть уже сейчас, то рекомендуется реализовать в прикладном коде временную функцию, которую при следующем обновлении библиотеки можно легко заменить на ее библиотечный эквивалент.

3.6. Другой пример скрытия деталей реализации библиотеки от потребителя. Допустим:

- в первой версии библиотеки потребителям предоставлялась экспортная функция общего модуля с повторным использованием возвращаемых значений;
- но в следующей версии библиотеки это проектное решение пересмотрено в пользу «обычного» общего модуля, куда эта функция была перенесена (аналогично, если в обратную сторону).

В данном примере, для того чтобы избавить потребителя библиотеки от дополнительных усилий по замене вызовов «старой» функции на новую, рекомендуется сразу размещать экспортную функцию в «обычном» модуле, в его разделе «программный интерфейс». Тогда эта функция, в зависимости от текущего проектного решения, может вызывать служебную функцию из модуля с повторным использованием возвращаемых значений или из любого другого модуля, или непосредственно сама содержать реализацию. Однако для потребителя ее местоположение уже не будет меняться в следующих версиях библиотеки.

4. Для упрощения контроля изменений программного интерфейса в новых версиях библиотек рекомендуется воспользоваться [приложенной обработкой](#).

Разработка ролей в библиотеках

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

#std668

1. При проектировании ролей в библиотеках рекомендуется придерживаться общего подхода к определению границ функциональности библиотеки:

- в библиотеку включаются те объекты метаданных, которые в неизменном или переопределенному виде войдут в состав конфигураций-потребителей.
- объекты, специфичные для конкретной конфигурации-потребителя, в библиотеке не разрабатываются, а создаются на этапе внедрения и настройки библиотеки в конфигурации.

2. В большинстве случаев, библиотека должна предоставлять готовый набор ролей для работы со своими данными (с объектами метаданных, содержащими данные: справочники, регистры и т.п.).

Например, вместе с универсальной подсистемой анкетирования в библиотеку входят роли для добавления и изменения шаблонов, опросов, ответов на вопросы анкет, просмотра ответов на вопросы и для администрирования (настройки) подсистемы анкетирования.

Иключение из этого правила составляют случаи, когда

- библиотечный объект не является «самостоятельным», а всегда подчинен какому-либо объекту-владельцу конфигурации. Например: библиотечные присоединенные файлы к объектам конфигурации (номенклатуре, проектам и т.п.) не имеют собственных ролей, т.к. доступ к файлам задается в ролях для работы с номенклатурой, проектами и т.п.
- библиотечный объект не является полностью законченным объектом, готовым для использования «как есть», а предполагает существенное переопределение в прикладных решениях. Например, библиотечные справочники **Организации**, **Физические лица** и т.п. содержат минимальную базовую реализацию, общую для всех прикладных решений, и не предоставляют «полномочие» роли для работы с ними.
- библиотека не содержит данных, а только алгоритмы для работы с ними.

3. При этом в ролях могут быть определены не только права к библиотечным объектам, но и логика ограничений доступа к ним на уровне записей (RLS – Row-Level Security).

В этом случае поставляемый RLS должен быть разработан таким образом, чтобы в прикладных решениях он использовался «как есть» и его не приходилось менять (дорабатывать). Тогда при обновлении библиотеки в конфигурации библиотечные роли всегда переносятся в конфигурацию «как есть».

Иключение из этого правила составляют случаи, когда

- в прикладном решении не предусмотрено никакой настройки прав доступа на уровне записей. В этом случае тексты RLS, если они предусмотрены, должны быть удалены из библиотечных ролей на этапе внедрения (обновления) библиотеки в конфигурацию.
- в прикладном решении предусмотрены прикладные сущности, в разрезе которых настраивается доступ, в том числе, к библиотечным объектам. Например, в разрезе «проектов» или «графов доступа» предусмотрена настройка видимости библиотечных «присоединенных файлов». В этом случае в тексты RLS в библиотечных ролях должна быть добавлена логика ограничения для этих сущностей на этапе внедрения (обновления) библиотеки в конфигурацию, либо должно быть принято решение не поставлять роли к таким объектам из библиотеки.

4. При использовании в библиотеке правил поставки для всех библиотечных ролей следует задавать правило поставки **«Изменения разрешены»**. Это требование обусловлено технологическими особенностями платформы 1С:Предприятие: при добавлении/удалении метаданных в конфигурации происходит неявно изменение ролей.

См. также

- [Стандартные роли](#)
- [Переопределяемые и поставляемые объекты библиотеки](#)
- [Настройка ролей и прав доступа](#)

Обработчики обновления информационной базы

Область применения: управляемое приложение, обычное приложение.

#std690

Действует для конфигураций на базе **Библиотеки стандартных подсистем**.

Содержит уточнения к требованиям других стандартов.

См. документацию к подсистеме [«Обновление версии ИБ»](#) на ИТС.

1. Основные сведения о библиотеке (основной конфигурации)
2. Расположение обработчиков обновления
3. Реализация обработчиков обновления
4. Переход на новые версии библиотек

1. Основные сведения о библиотеке (основной конфигурации)

1.1. При разработке конфигураций на базе библиотек, каждая библиотека должна сообщить о себе ряд сведений, необходимых для корректного обновления информационной базы на новую версию конфигурации:

- Имя
- Версию
- Список обработчиков обновления
- Зависимости от других библиотек.

Эти сведения размещаются в специальном общем модуле библиотеки, имя которого должно начинаться с **ОбновлениеИнформационнойБазы...**

Пример:

В конфигурации УТ 11 модуль обработчиков обновления называется **ОбновлениеИнформационнойБазыУТ**

Имена модулей всех используемых в конфигурации библиотек следует явно перечислить в общем модуле **ПодсистемыКонфигурацииПереопределяемый** в виде:

Процедура **При добавленииПодсистем(МодулиПодсистем)** Экспорт

МодулиПодсистем.Добавить("ОбновлениеИнформационнойБазыУТ");

КонецПроцедуры

Кроме того, аналогичный модуль также должен быть определен и для основной конфигурации.

При создании общего модуля **ОбновлениеИнформационнойБазы...** следует использовать шаблон:

```
////////////////////////////////////////////////////////////////  
// Обновление информационной базы <библиотеки или конфигурации>.  
//  
////////////////////////////////////////////////////////////////  
#Область ПрограммныйИнтерфейс  
////////////////////////////////////////////////////////////////  
// Получение сведений о библиотеке (или конфигурации).  
  
// Заполняет основные сведения о библиотеке или основной конфигурации.  
// Библиотека, имя которой имя совпадает с именем конфигурации в метаданных, определяется как основная конфигурация.  
//  
// Параметры:  
// Описание - Структура - сведения о библиотеке:  
//  
// Имя           - Страна - имя библиотеки, например, "СтандартныеПодсистемы".  
// Версия        - Страна - версия в формате из 4-х цифр, например, "2.1.3.1".  
//  
// ТребуемыеПодсистемы - Массив - имена других библиотек (Страна), от которых зависит данная библиотека.  
// Обработчики обновления таких библиотек должны быть вызваны ранее  
// обработчиками обновления данной библиотеки.  
//  
// При циклических зависимостях или, напротив, отсутствии каких-либо зависимостей,  
// порядок вызова обработчиков обновления определяется порядком добавления модулей  
// в процедуре ПриДобавленииПодсистем общего модуля ПодсистемыКонфигурацииПереопределляемый.  
// РежимВыполненияОтложенныхОбработчиков - Страна - "Последовательно" - отложенные обработчики обновления выполняются  
// последовательно в интервале от номера версии информационной базы до номера  
// версии конфигурации включительно или "Параллельно" - отложенный обработчик после  
// обработки первой порции данных передает управление следующему обработчику, а после  
// выполнения последнего обработчика цикл повторяется заново.  
//
```

Процедура ПриДобавленииПодсистемы(Описание) Экспорт

```
Описание.Имя      = "<Имя библиотеки>";  
Описание.Версия   = "XX.XX.XX.XX";  
Описание.ТребуемыеПодсистемы.Добавить("СтандартныеПодсистемы");  
Описание.РежимВыполненияОтложенныхОбработчиков = "Последовательно";
```

КонецПроцедуры

```
////////////////////////////////////////////////////////////////  
// Обработчики обновления информационной базы.
```

```
// Добавляет в список процедуры-обработчики обновления данных ИБ  
// для всех поддерживаемых версий библиотеки или конфигурации.  
// Вызывается перед началом обновления данных ИБ для построения плана обновления.  
//
```

```
// Параметры:  
// Обработчики - ТаблицаЗначений - описание полей  
//                   см. в процедуре ОбновлениеИнформационнойБазы.НоваяТаблицаОбработчиковОбновления  
//  
// Пример добавления процедуры-обработчика в список:  
// Обработчик = Обработчики.Добавить();  
// Обработчик.Версия      = "1.0.0.0";  
// Обработчик.Процедура    = "ОбновлениеИБ.ПерейтиНаВерсию_1_0_0_0";  
// Обработчик.РежимВыполнения = "Монопольно";  
//
```

Процедура ПриДобавленииОбработчиковОбновления(Обработчики) Экспорт

```
// Обработчики, выполняемые при каждом обновлении ИБ  
// Обработчики, выполняемые при переходе на определенную версию  
// Обработчики, выполняемые при заполнении пустой ИБ
```

КонецПроцедуры

```
// Вызывается перед процедурами-обработчиками обновления данных ИБ.  
//
```

Процедура ПередОбновлениемИнформационнойБазы() Экспорт

КонецПроцедуры

```
// Вызывается после завершения обновления данных ИБ.
```

```
// Параметры:  
// ПредыдущаяВерсия      - Страна - версия до обновления. "0.0.0.0" для "пустой" ИБ.  
// ТекущаяВерсия          - Страна - версия после обновления.  
// ВыполненныеОбработчики - ДеревоЗначений - список выполненных процедур-обработчиков обновления,  
//                                         сгруппированных по номеру версии.  
// ВыводитьОписаниеОбновлений - Булево - (возвращаемое значение) если установить Истина,  
//                                         то будет выведена форма с описанием обновлений. По умолчанию, Истина.  
// МонопольныйРежим       - Булево - Истина, если обновление выполнялось в монопольном режиме.  
//
```

Процедура ПослеОбновленияИнформационнойБазы(Знач ПредыдущаяВерсия, Знач ТекущаяВерсия,
Знач ВыполненныеОбработчики, ВыводитьОписаниеОбновлений, МонопольныйРежим) Экспорт

КонецПроцедуры

```
// Вызывается при подготовке табличного документа с описанием изменений в программе.
```

```
// Параметры:  
// Макет - ТабличныйДокумент - описание обновления всех библиотек и конфигурации.  
// Макет можно дополнить или заменить.  
// См. также общий макет ОписаниеИзмененийСистемы.
```

Процедура ПриПодготовкеМакетаОписанияОбновлений(Знач Макет) Экспорт

КонецПроцедуры

```
// Позволяет переопределить режим обновления данных информационной базы.
```

```
// Для использования в редких (нештатных) случаях перехода, не предусмотренных в  
// стандартной процедуре определения режима обновления.
```

```

// Параметры:
// РежимОбновленияДанных - Стока - в обработчике можно присвоить одно из значений:
//   "НачальноеЗаполнение" - если это первый запуск пустой базы (области данных);
//   "ОбновлениеВерсии" - если выполняется первый запуск после обновление конфигурации базы данных;
//   "ПереходСДругойПрограммы" - если выполняется первый запуск после обновление конфигурации базы данных,
//   в которой изменилось имя основной конфигурации.
//
// СтандартнаяОбработка - Булево - если присвоить Ложь, то стандартная процедура
// определения режима обновления не выполняется,
// а используется значение РежимОбновленияДанных.
//
Процедура ПриОпределенииРежимаОбновленияДанных(РежимОбновленияДанных, СтандартнаяОбработка) Экспорт

КонецПроцедуры

// Добавляет в список процедуры-обработчики перехода с другой программы (с другим именем конфигурации).
// Например, для перехода между разными, но родственными конфигурациями: базовая -> проф -> корп.
// Вызывается перед началом обновления данных ИБ.
//
// Параметры:
// Обработчики - ТаблицаЗначений - с колонками:
//   * ПредыдущееИмяКонфигурации - Стока - имя конфигурации, с которой выполняется переход;
//   или "*", если нужно выполнять при переходе с любой конфигурации.
//   * Процедура - Стока - полное имя процедуры-обработчика перехода с программы ПредыдущееИмяКонфигурации.
//   Например, "ОбновлениеИнформационнойБазыУПП.ЗаполнитьУчетнуюПолитику"
//   Обязательно должна быть экспортной.
//
// Пример добавления процедуры-обработчика в список:
// Обработчик = Обработчики.Добавить();
// Обработчик.ПредыдущееИмяКонфигурации = "УправлениеТорговлей";
// Обработчик.Процедура = "ОбновлениеИнформационнойБазыУПП.ЗаполнитьУчетнуюПолитику";
//
Процедура ПриДобавленииОбработчиковПереходаСДругойПрограммы(Обработчики) Экспорт

КонецПроцедуры

// Вызывается после выполнения всех процедур-обработчиков перехода с другой программы (с другим именем конфигурации),
// и до начала выполнения обновления данных ИБ.
//
// Параметры:
// ПредыдущееИмяКонфигурации - Стока - имя конфигурации до перехода.
// ПредыдущаяВерсияКонфигурации - Стока - имя предыдущей конфигурации (до перехода).
// Параметры - Структура -
//   * ВыполнитьОбновлениеСВерсии - Булево - по умолчанию Истина. Если установить Ложь,
//   то будут выполнена только обязательные обработчики обновления (с версией "*").
//   * ВерсияКонфигурации - Стока - номер версии после перехода.
//   По умолчанию, равен значению версии конфигурации в свойствах метаданных.
//   Для того чтобы выполнить, например, все обработчики обновления с версии ПредыдущаяВерсияКонфигурации,
//   следует установить значение параметра в ПредыдущаяВерсияКонфигурации.
//   Для того чтобы выполнить вообще все обработчики обновления, установить значение "0.0.0.1".
//   * ОчиститьСведенияоПредыдущейКонфигурации - Булево - по умолчанию Истина.
//   Для случаев когда предыдущая конфигурация совпадает по имени с подсистемой текущей конфигурации, следует указать Ложь.
//
Процедура ПриЗавершенииПереходаСДругойПрограммы(Знач ПредыдущееИмяКонфигурации,
Знач ПредыдущаяВерсияКонфигурации, Параметры) Экспорт
```

КонецПроцедуры

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

```
///////////////////////////////
// Заполнение пустой ИБ
///////////////////////////////
// Обновление ИБ
/////////////////////////////
```

#КонецОбласти

1.2. Обработчики обновления данных информационной базы предназначены для дополнительной обработки данных после завершения обновления конфигурации (реструктуризации) базы данных:

- инициализация новых констант, новых реквизитов, реквизитов новых предопределенных элементов;
- перенос данных из устаревших структур метаданных в новые;
- генерация новых данных
- и т.п.

Для автогенерируемых строк, которые программно записываются в информационную базу, например при заполнении наименований предопределенных элементов справочников, ПВХ и т.п., следует руководствоваться стандартом [Автогенерированные данные в информационной базе: требования по локализации](#).

1.3. Обработчик обновления данных информационной базы состоит из двух частей:

- описательной - сообщает, когда должен выполниться обработчик, и где он находится в конфигурации;
- программной - непосредственно код модификации данных ИБ, оформленный в виде процедуры-обработчика обновления.

Добавление описаний новых обработчиков выполняется в процедуре **ПриДобавленииОбработчиковОбновления** с помощью вставки фрагмента кода по шаблону:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "<номер версии>";
Обработчик.Процедура = "<полное имя экспортной процедуры>";
Обработчик.НачальноеЗаполнение = {Истина|Ложь};
Обработчик.РежимВыполнения = {"Монопольно"|"Оперативно"|"Отложенно"};
```

Данный код добавляет новую строку в таблицу значений **Обработчики**, строка которой имеет следующие поля:

Версия (Строка) – номер версии конфигурации, при обновлении на которую должна быть вызвана процедура обновления, указанная в параметре **Процедура**.

- Номер версии конфигурации указывается в формате «Р.П.В.С» (Р – старший номер редакции; П – младший номер редакции; В – номер версии; С – номер сборки. Если следующую версию нельзя определить, то можно указать следующий номер сборки).
- Если в качестве версии указан символ «*», то обработчик обновления должен выполняться каждый раз при обновлении информационной базы, независимо от номера версии конфигурации. Обработчики такого вида предназначены для обновления служебных, системных данных (например, обновление поставляемых профилей и групп доступа).
- Если свойство **Версия** не задано, то должно быть установлено в **Истина** свойство **НачальноеЗаполнение** (см. далее).

Процедура (Строка) – идентификатор процедуры, содержащий полный путь к процедуре-обработчику обновления. Например, "Справочник.Валюты.ЗаполнитьКодДляПоиска".

Начальное заполнение (Булево) – если **Истина**, то обработчик будет вызван при первом запуске пустой информационной базы (версия «0.0.0.0»), созданной из файла поставки конфигурации и не содержащей данных.

Это обработчики первоначального заполнения базы.

Режим выполнения (Строка) – принимает одно из значений: "Монопольно", "Оперативно" и "Отложенно". Если свойство не задано, то по умолчанию обработчик – монопольный.

- **Монопольно** – если обработчик обновления необходимо выполнять монопольно, в условиях отсутствия активных сеансов работы пользователей, регламентных заданий, внешних соединений и подключений по веб-сервисам. В противном случае, обновление версии программы прерывается. Подробнее см. Ограничения на использование монопольного режима обработчиков обновления

Монопольные обработчики предназначены для обновления тех данных, обработка которых должна быть обязательно завершена к моменту входа пользователей в программу. Для сокращения времени простоя (ожидания обработки данных), рекомендуется большие объемы данных обновлять отложенно (см. ниже).

Примеры монопольных обработчиков: обработка небольшого объема данных текущего периода, активных позиций номенклатуры и т.п.

Если хотя бы один обработчик обновления конфигурации – монопольный, то все оперативные обработчики (см. далее) выполняются в монопольном режиме.

В случае если обработчик обновления – обязательный (свойство Версия = «*»), то значение **Монопольно** следует устанавливать только в тех случаях, когда обработчик обновления должен программно определить, требуется ли монопольный режим для его выполнения:

- Такой обработчик вызывается дважды, в него передается параметр **Параметры типа Структура**, в котором имеется свойство **МонопольныйРежим** (Булево)
- При первом вызове в режиме проверки, свойство **МонопольныйРежим** содержит значение **Ложь**.
 - Код обработчика не должен модифицировать данные ИБ
 - Если в ходе выполнения обработчика возникает необходимость внесения изменений в ИБ, обработчик должен установить значение свойства в **Истина** и прекратить свое выполнение
- При втором вызове в режиме выполнения свойство **МонопольныйРежим** содержит значение **Истина**
 - Код обработчика может модифицировать данные ИБ
 - Изменение значения свойства в этом случае игнорируется

- **Оперативно** – если обработчик обновления необходимо выполнять не монопольно: при активных сеансах работы пользователей, регламентных заданий, внешних соединений и подключений через веб-сервисы.

Оперативные обработчики следуют применять в редких случаях, когда важно сократить время ожидания пользователей при переходе на исправительные релизы, которые не содержат изменений в структуре данных, и обновление на которые должно выполняться динамически.

Подробнее см. *Оперативное обновление данных*.

- **Отложенно** – если обработчик обновления необходимо выполнять в фоне после того, как завершено выполнение монопольных (оперативных) обработчиков, и пользователям уже разрешен вход в программу.

Отложенные обработчики предназначены для обработки той части данных ИБ, которые не препятствуют пользователям начинать свою работу с новой версией программы, не дожидаясь завершения обработки этих данных.

Примеры отложенных обработчиков: обработка больших архивов данных за закрытые/прошлые периоды, неактивных позиций номенклатуры, различных данных, отключенных в данный момент функциональными опциями и т.п.

Подробнее см. *Отложенное обновление данных*.

Если в конфигурации (библиотеке) используется параллельный режим отложенного обновления (в процедуре **ПриДобавленииПодсистемы** свойство **РежимВыполненияОтложенныхОбработчиков** = "Параллельно"), то для написания отложенных обработчиков следует руководствоваться стандартом *Параллельный режим отложенного обновления*.

Пример описания обработчика, для выполнения которого требуется монопольный режим:

```
Обработчик = Обработчики.Добавить();  
Обработчик.РежимВыполнения = "Монопольно";  
Обработчик.Версия = "11.1.0.0";  
Обработчик.Процедура = "Справочник.МойСправочник.ЗаполнитьКодДляПоиска";
```

Пример реализации обработчика в модуле менеджера **Справочник.МойСправочник**:

```
// Обработчик обновления УТ 11.1.0.0  
//  
// Перебираются все элементы справочника, в которых не заполнен код для поиска,  
// и заполняется кодом справочника без лидирующих нулей и префиксов  
//  
Процедура ЗаполнитьКодДляПоиска() Экспорт  
...
```

2. Расположение обработчиков обновления

2.1. Процедура-обработчик должна оформляться в виде экспортной процедуры.

Располагать процедуру следует в модуле менеджера того объекта метаданных, обновление которого она выполняет.

Пример:

Если в справочник «Подразделения» добавили новый реквизит, который необходимо заполнить значением по умолчанию, то процедура-обработчик должна располагаться в модуле менеджера этого справочника.

2.2. В некоторых случаях, когда невозможно соотнести обработчик с каким-то конкретным объектом метаданных, допустимо расположение процедуры-обработчика в серверном общем модуле, назначение которого по смыслу связано с выполняемой обработкой ИБ (например, процедуры обновления, связанные со складской функциональностью должны располагаться в общем модуле СкладСервер). При этом процедура должна располагаться в служебной части модуля, в подразделе «Обновление ИБ».

3. Реализация обработчиков обновления

3.1. К процедуре-обработчику предъявляются следующие требования:

- Обработчик не должен содержать логики по интерактивному взаимодействию с пользователем.
- В случае критической ошибки при обновлении, в обработчике необходимо вызывать исключение, которое приведет к остановке всей процедуры обновления. Остановка обновления информационной базы приведет к невозможности запуска до тех пор, пока причины ошибки не будут устранены.
- Обработчик должен быть рассчитан на неоднократное выполнение. На одних и тех же данных результат выполнения обработчика должен быть идентичен при любом количестве вызовов этого обработчика (например, повторный запуск обработчика не должен приводить к дублированию данных в информационной базе).
- В пределах одной версии (значение свойства **Версия**), работоспособность обработчика не должна ставиться в зависимость от очередности его выполнения. Если подобные зависимости проявляются, то такие обработчики необходимо объединять в один.
- Если в конфигурации предусмотрены [планы обмена РИБ с отборами](#), то также нужно учитывать, что в обновляемом подчиненном узле РИБ могут быть неполные данные: например, в нем имеются движения по регистру, а сам регистратор отсутствует. При этом обработчик обновления должен пропускать обновление таких данных.

3.2. Обработчик обновления не должен содержать лишних, избыточных действий с данными – должен выполняться максимально быстро.

3.2.1. Для этого необходимо отключать бизнес-логику при обработке данных. В большинстве случаев, с помощью установки признака **ОбменДанными.Загрузка**:

```
ДокументОбъект.ОбменДанными.Загрузка = Истина;
```

В отдельных случаях, для частичного отключения бизнес-логики допустимо предусмотреть дополнительный признак, например:

ДокументОбъект.ДополнительныеСвойства.Вставить("ОтключитьМоюБизнесЛогику");

3.2.2. Для большинства обрабатываемых данных следует отключать регистрацию изменений на узлах планов обмена, чтобы избежать отправки всего объема обработанных данных во все узлы. Таким образом:

- В распределенной информационной базе (РИБ) обработка данных должна выполняться независимо в каждом из узлов;
- При обмене между произвольными конфигурациями (программами) обработка данных не должна приводить к их выгрузке в базы-получатели.

Исключение составляют случаи создания ссылочных объектов, которые должны быть перенесены механизмами обмена данными в другие узлы РИБ с тем же значением реквизита Ссылка.

3.2.3. Таким образом, в коде обработчика обновления вместо

ДокументОбъект.Записать();

должно быть:

```
ДокументОбъект.ОбменДанными.Загрузка = Истина; // отключить всю бизнес-логику при записи
ДокументОбъект.ДополнительныеСвойства.Вставить("ОтключитьМеханизмРегистрацииОбъектов");
ДокументОбъект.ОбменДанными.Получатели.АвтоЗаполнение = Ложь;
ДокументОбъект.Записать();
```

При использовании в конфигурации Библиотеки стандартных подсистем (БСП) версии 2.1.4 и выше следует использовать процедуру **ЗаписатьДанные** общего модуля **ОбновлениеИнформационнойБазы**:

ОбновлениеИнформационнойБазы.ЗаписатьДанные(ДокументОбъект);

3.3. Перед процедурой-обработчиком должен быть комментарий. При этом первая строка комментария должна содержать информацию о версии конфигурации, для которой предназначен этот обработчик. Последующие строки комментария должны содержать ответ на следующие вопросы:

- Какие данные будут изменены (что меняем)?
- Какие изменения будут внесены в эти данные (как меняем)?

Пример:

```
// Обработчик обновления УТ 11.1.0.0
//
// Перебираются все элементы справочника, в которых не заполнен код для поиска,
// и заполняется кодом справочника без лидирующих нулей и префиксов
//
Процедура ЗаполнитьКодДляПоиска() Экспорт
```

4. Переход на новые версии библиотек

4.1. При постановке конфигурации на поддержку к новой версии библиотеки, следует увеличивать номер версии конфигурации. Это необходимо для запуска обработчиков обновления информационной базы.

См. также

- [Начальные действия при работе конфигурации](#)

Общие требования по локализации конфигурации

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

#std458

При разработке следует ориентироваться на то, что

- в конфигурации предусмотрено несколько языков интерфейса пользователя,
- данные в базе данных вводятся на одном языке, отличном от основного языка конфигурации (см. свойство **ОсновнойЯзык** метаданных).

При этом не регламентируются

- ввод данных сразу на нескольких языках,
- многоязычное представление метаданных, хранящихся в данных (например, предопределенные элементы, дополнительные реквизиты и т.п.).

См. также

- [Поставка международной версии конфигурации](#)
- [Интерфейсные тексты в коде: требования по локализации](#)
- [Запросы, динамические списки и отчеты на СКД: требования по локализации](#)
- [Форматирование даты, числа, Булево: требования по локализации](#)
- [Строковые константные выражения в коде: требования по локализации](#)
- [Элементы форм: требования по локализации](#)
- [Регламентные задания: требования по локализации](#)
- [Макеты: требования по локализации](#)
- [Автогенерированные данные в информационной базе: требования по локализации](#)

Поставка международной версии конфигурации

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

#std769

Применим для локализуемых конфигураций (и входящих в них библиотек), на базе которых выпускаются национальные прикладные решения для различных стран или регионов.

Для упрощения работы локализаторов при первичной локализации конфигурации, а также при последующих обновлениях следует выпускать международную версию конфигурации, подготовленную для создания национальных версий. При этом разработку локализованных версий конфигурации рекомендуется вести на базе международной версии.

Все требования данного раздела (группа стандартов **Требования по локализации**) одинаково применимы к обоим версиям конфигурации. В частности, национальные версии также могут поставляться с интерфейсами на нескольких языках. Например, ERP для России с англоязычным интерфейсом для работы иностранцев в одной информационной базе с русскоговорящими пользователями.

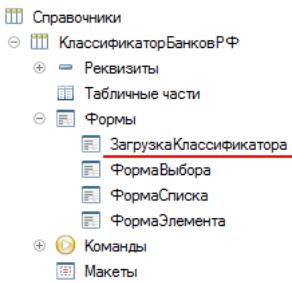
С целью уменьшения трудозатрат на выделение национальной специфики следует анализировать в первую очередь зарубежные версии конфигураций, опубликованные на [портале 1С](#).

Для подготовки международной версии конфигурации:

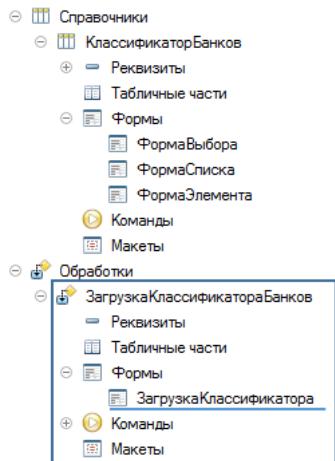
1. Национальную (российскую) специфику в программном коде и формах конфигурации необходимо выделить в отдельные объекты метаданных, которые должны отсутствовать в международной версии.

Например,

неправильно:

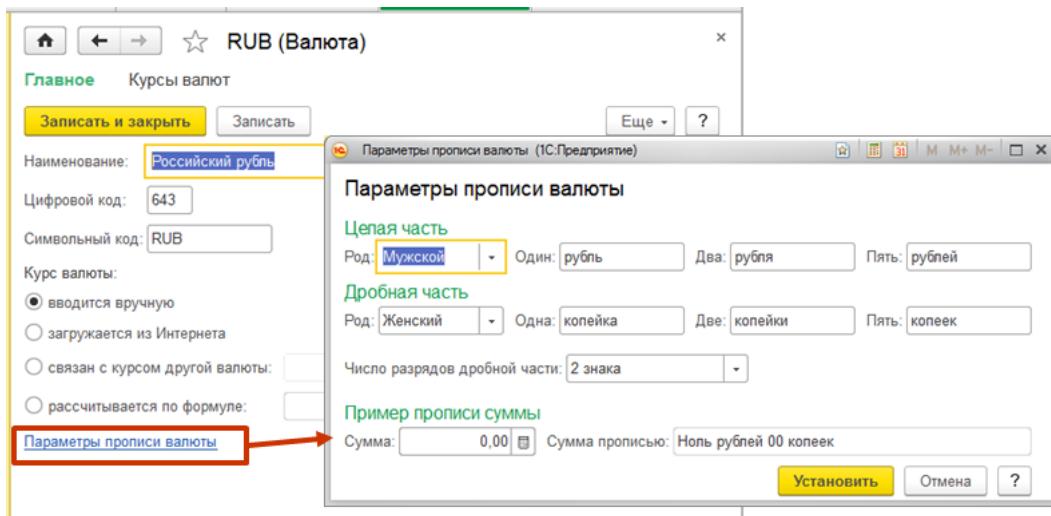


правильно:



1.1. Реквизиты объектов, относящиеся к национальной специфике, рекомендуется оставлять в объектах, но при этом обеспечить работоспособность конфигурации в случае их удаления.

1.2. В формах объектов, где к национальной специфике относится лишь часть элементов, рекомендуется выносить такие элементы в отдельную форму, например:



1.3. К национальной специфике могут относиться и алгоритмы, например: загрузка курсов валют. Их также следует выделять в отдельных объектах метаданных.

1.4. В случае, когда подсистема или отдельный объект в большей части является международным, но при этом очевидна необходимость реализации некоторой национальной специфики, рекомендуется создавать общие модули с постфиксом **Локализация**. В алгоритмах, в которых есть потребность дополнить или переопределить поведение, предусмотреть переход в эти модули. При этом в международной поставке модули с постфиксом **Локализация** не должны содержать прикладного кода.

Например, функциональность закрытия месяца является международной, но при этом есть необходимость дополнять список этапов теми этапами, которые реализуют национальную специфику.

В этом случае добавляется общий модуль **ЗакрытиеМесяцаЛокализация**, а из процедуры **ЗакрытиеМесяца.ДобавитьЭтапыЗакрытияМесяца** вызывается соответствующий метод модуля **ЗакрытиеМесяцаЛокализация**. При этом в международной поставке конфигурации общий модуль **ЗакрытиеМесяцаЛокализация** будет содержать только определение метода **ДобавитьЭтапыЗакрытияМесяца**.

1.5. В случае, когда национальная специфика занимает большую часть подсистемы, рекомендуется относить к национальной специфике всю подсистему. Например: подсистема «Склонение представлений объектов» целиком является национальной спецификой.

1.6. Перечислить объекты метаданных, содержащих национальную (российскую) специфику, в файле **ЛокализуемыеОбъекты<...>.txt**, и включить его в дистрибутив типовой конфигурации.

Файл **ЛокализуемыеОбъекты<...>.txt** содержит имена объектов метаданных в том виде, в котором их возвращает функция **ОбъектМетаданных.ПолноеИмя()**. Каждое имя объекта начинается с новой строки. Допускаются комментарии в таком же виде, как и в коде конфигурации.

Пример:

```
// Банки  
Обработка.ЗагрузкаКлассификатораБанков  
РегламентноеЗадание.ЗагрузкаКлассификатораБанков  
Справочник.КлассификаторБанков.Реквизит.ИНН  
  
// Валюты
```

Обработка.ЗагрузкаКурсовВалют Регламентное Задание.ЗагрузкаКурсовВалют

2. Удобство пользовательского интерфейса не должно страдать при соблюдении требования №1. Если юзабилити формы значительно ухудшается вследствие переноса её части в отдельные формы, в этом случае рекомендуется всю форму относить к национальной специфике.

3. В процедурах переопределяемых модулей допустимо помещать только вызовы процедур конфигурации, содержащих программный код. Это упрощает доработку переопределяемых модулей при локализации. Например,

неправильно:

// В модуле ВариантыОтчетовПереопределяемый

```
Процедура ОпределитьРазделыСВариантамиОтчетов(Разделы) Экспорт
    Разделы.Добавить(ВариантыОтчетовКлиентСервер.ИдентификаторНачальнойСтраницы(), 
        НСтр("ru = 'Главное'"));

    Разделы.Добавить(Метаданные.Подсистемы.CRMIMаркетинг,
        НСтр("ru= 'Отчеты по CRM и маркетингу'"));

    Разделы.Добавить(Метаданные.Подсистемы.Закупки,
        НСтр("ru= 'Отчеты по закупкам'"));

    Разделы.Добавить(Метаданные.Подсистемы.Казначейство,
        НСтр("ru= 'Отчеты по казначейству'"));

    Разделы.Добавить(Метаданные.Подсистемы.Продажи,
        НСтр("ru= 'Отчеты по продажам'"));

    Разделы.Добавить(Метаданные.Подсистемы.Склад,
        НСтр("ru= 'Отчеты по складу'"));

    Разделы.Добавить(Метаданные.Подсистемы.ФинансовыйРезультатИКонтроллинг,
        НСтр("ru= 'Отчеты по финансовому результату'"));

КонецПроцедуры
```

правильно:

// В модуле ВариантыОтчетовПереопределяемый

```
Процедура ОпределитьРазделыСВариантамиОтчетов(Разделы) Экспорт
    ВариантыОтчетовУТ.ОпределитьРазделыСВариантамиОтчетов(Разделы);
КонецПроцедуры
```

// В модуле ВариантыОтчетовУТ

```
Процедура ОпределитьРазделыСВариантамиОтчетов(Разделы) Экспорт
    Разделы.Добавить(ВариантыОтчетовКлиентСервер.ИдентификаторНачальнойСтраницы(),
        НСтр("ru = 'Главное'"));

    Разделы.Добавить(Метаданные.Подсистемы.CRMIMаркетинг,
        НСтр("ru= 'Отчеты по CRM и маркетингу'"));

    Разделы.Добавить(Метаданные.Подсистемы.Закупки,
        НСтр("ru= 'Отчеты по закупкам'"));

    Разделы.Добавить(Метаданные.Подсистемы.Казначейство,
        НСтр("ru= 'Отчеты по казначейству'"));

    Разделы.Добавить(Метаданные.Подсистемы.Продажи,
        НСтр("ru= 'Отчеты по продажам'"));

    Разделы.Добавить(Метаданные.Подсистемы.Склад,
        НСтр("ru= 'Отчеты по складу'"));

    Разделы.Добавить(Метаданные.Подсистемы.ФинансовыйРезультатИКонтроллинг,
        НСтр("ru= 'Отчеты по финансовому результату'"));

КонецПроцедуры
```

4. Необходимо обеспечивать работоспособность международной конфигурации.

5. Следует выпускать международную и российскую версии конфигурации синхронно. Для этого рекомендуется выбрать одну из схем разработки:

a. Вести разработку российской версии в одном хранилище разработки. Автоматически изготавливать международную версию из российской при сборке дистрибутива путем удаления объектов метаданных, содержащих российскую специфику, и удалением кода внутри процедур общих модулей с постфиксом **Локализация**.

b. Вести разработку российской и международной версии в двух хранилищах разработки, поставив российскую версию на поддержку к международной и добавив объекты метаданных, содержащих российскую специфику. Реализацию российской специфики вести в добавляемых объектах и модулях с постфиксом **Локализация**.

При этом локализаторам рекомендуется вести разработку своих локализованных версий конфигурации на базе международной версии. Для этого:

- поставить свою национальную конфигурацию на поддержку международной;
- по мере необходимости добавить в конфигурацию объекты, перечисленные в файле ЛокализуемыеОбъекты<...>.txt из поставки типовой конфигурации, взяв в качестве образца аналогичные объекты в российской версии конфигурации;
- при необходимости вписать реализацию в процедуры общих модулей с постфиксом **Локализация**.

Но если за основу берется российская версия конфигурации, то локализацию рекомендуется выполнять следующим образом:

- поставить свою национальную конфигурацию на поддержку к российской;
- снять с поддержки и локализовать объекты, перечисленные в файле ЛокализуемыеОбъекты<...>.txt из поставки типовой конфигурации;
- снять с поддержки и локализовать алгоритмы общих модулей с постфиксом **Локализация**.

Подготовку файла поставки международной версии можно упростить, используя файл настроек объединения.

Для автоматизации предусмотрена [обработка подготовки файла настроек объединения](#).

См. также

- [Интерфейсные тексты в коде: требования по локализации](#)
- [Запросы, динамические списки и отчеты на СКЛ: требования по локализации](#)
- [Даты: требования по локализации](#)
- [Строковые константные выражения в коде: требования по локализации](#)
- [Элементы форм: требования по локализации](#)
- [Регламентные задания: требования по локализации](#)

- [Макеты: требования по локализации](#)

Интерфейсные тексты в коде: требования по локализации

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

#std761

1. Недопустимо выводить в пользовательском интерфейсе, в сообщениях для пользователя, справочной информации и других материалах имена объектов метаданных и внутренние идентификаторы, которые используются в коде. В пользовательском интерфейсе допустимо выводить синонимы (представления) объектов метаданных и представления внутренних идентификаторов.
При этом рекомендуется получать синонимы напрямую из метаданных с помощью функции **Представление**, а не дублировать в текстах. Это исключит ошибки перевода, когда переводчики могут переводить один и тот же термин разными способами.

Например, неправильно:

```
СообщениеОшибка = НСтр("ru='Обнаружены отрицательные остатки по регистру ТоварыОрганизаций.'");
```

Правильно:

```
СообщениеОшибка = СтрШаблон(НСтр("ru = 'Обнаружены отрицательные остатки по регистру \"%1\".'"),  
Метаданные.РегистрыНакопления.ТоварыОрганизаций.Представление());
```

Или сокращенный вариант:

```
СообщениеОшибка = СтрШаблон(НСтр("ru = 'Обнаружены отрицательные остатки по регистру \"%1\".'"),  
Метаданные.РегистрыНакопления.ТоварыОрганизаций);
```

1.1. Исключение составляют сообщения для разработчиков, интерфейсы различных инструментов разработчиков или внедренцев, для которых предусмотрен вывод имен объектов метаданных и внутренних идентификаторов. При выводе в сообщениях имен процедур и функций следует для однозначности выводить их полные имена с именем общего модуля или модуля менеджера.

Неправильно:

```
СообщениеОшибка = НСтр("ru = 'Ошибка в функции ЕстьРоль модуля Управлениедоступом.'");
```

Правильно:

```
ИмяФункции = "Управлениедоступом.ЕстьРоль";  
СообщениеОшибка = СтрШаблон(НСтр("ru = 'Ошибка в функции %1.'"), ИмяФункции);
```

Пример для модуля менеджера:

```
ИмяФункции = "Документы.АвансовыйОтчет.АдаптированныйТекстЗапросаДвиженийПоРегистру";  
СообщениеОшибка = СтрШаблон(НСтр("ru = 'Для исправления движений документа необходимо вызвать функцию %1.'"), ИмяФункции);
```

При выводе имен объектов метаданных рекомендуется получать их напрямую из метаданных с помощью функции **ПолноеИмя**. Это исключит ошибки перевода, когда переводчики могут пытаться переводить заранее непереводимые имена объектов метаданных и внутренних идентификаторов.

Например, неправильно:

```
СообщениеОшибка = НСтр("ru='Подсистема некорректно встроена в регистр ТоварыОрганизаций.'");
```

Правильно:

```
СообщениеОшибка = СтрШаблон(НСтр("ru = 'Подсистема некорректно встроена в регистр %1.'"),  
Метаданные.РегистрыНакопления.ТоварыОрганизаций.ПолноеИмя));
```

2. Если в модулях конфигурации встречаются строки, предназначенные для пользовательского интерфейса (сообщения пользователю, надписи в формах, названия и подсказки команд, выражения в настройках СКДи т.п.) необходимо обеспечить возможность локализации таких строк.

Для этого необходимо применять функцию **НСтр** вместо прямого использования строковых литералов. Иное использование строк, предназначенных для пользовательского интерфейса, не допускается.

Например, неправильно:

```
ПоказатьПредупреждение(, "Для выполнения операции необходимо установить расширение работы с файлами.");
```

Правильно:

```
ПоказатьПредупреждение(, НСтр("ru='Для выполнения операции необходимо установить расширение работы с файлами.'"));
```

Также следует обращать внимание на корректное использование функции **НСтр**.

Например, неправильно:

```
ТекстСообщения = "ru='Для выполнения операции необходимо установить расширение работы с файлами.'";  
ПоказатьПредупреждение(, НСтр(ТекстСообщения));
```

Правильно:

```
ТекстСообщения = НСтр("ru='Для выполнения операции необходимо установить расширение работы с файлами.'");  
ПоказатьПредупреждение(, ТекстСообщения);
```

3. В том случае если строка является составной и включает в себя части, зависящие от тех или иных условий, тем не менее, следует использовать логически завершенные, целостные фразы (предложения). Необходимо применять функцию **СтрШаблон** (или аналогичную) для подстановки параметров в строки сообщений пользователю.

Это требование обусловлено, во-первых, разным расположением параметров в тексте предложения на различных языках, что приводит к необходимости изменения исходного кода для подстановки складываемых фрагментов строк, а во-вторых, невозможностью корректно перевести отдельные несогласованные части предложения (наличие или отсутствие артиклей, предлогов, разное расположение знаков препинания в тексте предложения на различных языках и т.п.).

Неправильно:

```
Сообщение0Нхватке = "Не хватает товара " + НаименованиеТовара + " на складе " + НаименованиеСклада + ".";
```

Правильно:

```
ТекстСообщения = НСтр("ru = 'Не хватает товара %1 на складе %2.'");  
Сообщение0Нхватке = СтрШаблон(ТекстСообщения, НаименованиеТовара, НаименованиеСклада);
```

Также с помощью параметров подстановки не следует разрывать целостную фразу на отдельные логически незавершенные части. Вместо этого следует задавать в коде несколько строк со всеми вариантами фразы.

Неправильно:

```
НСтр("ru = '%1 пользователя \"%2\" в группу \"%3\"?!" // где параметр %1 может принимать слова "Включить", "Копировать" или "Удалить
```

Правильно:

```
НСтр("ru = 'Включить пользователя """%2"" в группу """%3""?'''")
НСтр("ru = 'Копировать пользователя """%2"" в группу """%3""?'''")
НСтр("ru = 'Удалить пользователя """%2"" в группу """%3""?'''")
```

В то же время допустимым является:

1. Составление текста из нескольких предложений (каждое из которых заключено в **НСтр** и переводится отдельно).
2. Предложения, заканчивающиеся двоеточием. Например, НСтр("ru = 'Создание каталога не выполнено по причине:'").

В связи со сложившейся практикой, допускается использовать именованные параметры подстановки (параметры, включающие имя аналогично переменной, а не номер) только в двух вариантах: [Параметр], %Параметр%. Здесь Параметр должен удовлетворять требованиям стандарта [Правила образования имен переменных](#).

Правильно:

```
Сообщение0Нехватке = НСтр("ru='Не хватает товара %Товар% на складе %Склад%.')")
Сообщение0Нехватке = СтрЗаменить(Сообщение0Нехватке, "%Товар%", НаименованиеТовара);
Сообщение0Нехватке = СтрЗаменить(Сообщение0Нехватке, "%Склад%", НаименованиеСклада);
```

4. При использовании в конфигурации Библиотеки стандартных подсистем для составных форматированных строк вместо объекта **ФорматированнаяСтрока** следует применять функцию **ФорматированнаяСтрока** общих модулей **СтроковыеФункции** или **СтроковыеФункцииКлиент**.

Неправильно:

```
Текст = Новый Массив;
Текст.Добавить(НСтр("ru = 'Перед удалением расширения рекомендуется'"));
Текст.Добавить(" ");
Текст.Добавить(Новый ФорматированнаяСтрока(НСтр("ru = 'выполнить резервное копирование информационной базы.'"),
ШрифтыСтиля.ПолужирныйШрифт));
ТекстПредупреждения = Новый ФорматированнаяСтрока(Текст);
```

Правильно:

```
ТекстПредупреждения = СтроковыеФункцииКлиент.ФорматированнаяСтрока(НСтр("ru = 'Перед удалением расширения рекомендуется <b>выполнить</b> резервное копирование информационной базы'."));
```

5. В функции **НСтр** строка ограничивается символами одинарных кавычек. Такое требование обусловлено частым использованием двойных кавычек в строковых литералах, а также встроенным в платформу механизмом редактирования строк на разных языках.

Неправильно:

```
ПоказатьПредупреждение(, НСтр("ru=Переменная типа ""Строка"""));
ПоказатьПредупреждение(, НСтр("ru=""Переменная типа ""Строка"""""));
```

Правильно:

```
ПоказатьПредупреждение(, НСтр("ru='Переменная типа ""Строка""'"));
```

6. При использовании функций **ЧислоПрописью**, **ПредставлениеПериода**, **СтрокаСЧислом** не следует указывать параметр "Л="("L=") в строке форматирования.

Неправильно:

```
СуммаПрописью = ЧислоПрописью(2341.56, "Л = ru_RU; ДП = Истина", НСтр("ru='доллар,доллара,долларов,м,цент,цента,центов,м,2'"));
```

Правильно:

```
СуммаПрописью = ЧислоПрописью(2341.56, "ДП = Истина", НСтр("ru='доллар,доллара,долларов,м,цент,цента,центов,м,2'"));
```

7. В редких случаях, например, когда нужно собрать длинное сообщение с предоставлением лога действий пользователя, допускается применять не замену строк в строке-шаблоне, а сложение строк. При этом неязыковые символы (чаще перенос строки) в начале и конце строк необходимо выделять в отдельные строковые литералы (которые пропускаются при переводе).

Неправильно:

```
ТекстСообщения = НСтр("ru = 'Не удалось сохранить файл документа по причине:
| ''')
+ ИнформацияОб ошибке;
```

Правильно:

```
ТекстСообщения = НСтр("ru = 'Не удалось сохранить файл документа по причине:'")
+ Символы.ПС + ИнформацияОб ошибке;
```

В противном случае, при переводе строки на другой язык концевой пробел легко может быть не замечен переводчиком, так как переводчик не видит всего контекста, а только сводную таблицу строк, подлежащих переводу. Кроме того, может быть искажена при переводе фраза, так как её продолжение переводчику не видно и нет символа шаблона подстановки, поясняющего, что дальше будет продолжение.

8. При вызове метода **ПоказатьВопрос** с указанием кнопок диалога в параметре **Кнопки**:

- следует по возможности использовать системное перечисление **КодВозвратаДиалога**;
- Если в перечислении нет нужной кнопки, вместе со значением, связанным с кнопкой, следует задавать его представление с использованием функции **НСтр**.

Например, неправильно:

```
Кнопки = Новый СписокЗначений;
Кнопки.Добавить("Отключить");
Кнопки.Добавить("Нет");
ПоказатьВопрос(..., Кнопки);
```

Правильно:

```
Кнопки = Новый СписокЗначений;
Кнопки.Добавить("Отключить", НСтр("ru='Отключить'"));
Кнопки.Добавить(КодВозвратаДиалога.Нет);
ПоказатьВопрос(..., Кнопки);
```

См. также

- [Использование Журнала регистрации](#)

Запросы, динамические списки и отчеты на СКД: требования по локализации

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

#std762

1. В некоторых случаях строковые литералы из текстов запросов также могут оказаться частью пользовательского интерфейса. В таких случаях строковые литералы необходимо выносить из текста запроса в параметры.

Неправильно:

```

ЗапросПоВерсиям = Новый Запрос()
| ВЫБРАТЬ
| Версии.Ссылка,
| ВЫБОР КОГДА Версии.Выпущена = ИСТИНА
| | ТОГДА "'выпущена'"
| | ИНАЧЕ "'в разработке'"
| КОНЕЦ КАК ТекстПояснения
| ИЗ
| Справочник.Версии КАК Версии");

```

Также неправильно:

```

ТекстЗапроса =
"ВЫБРАТЬ
| Версии.Ссылка,
| ВЫБОР КОГДА Версии.Выпущена = ИСТИНА
| | ТОГДА &ТекстВыпущенойВерсии
| | ИНАЧЕ &ТекстНевыпущенойВерсии
| КОНЕЦ КАК ТекстПояснения
| ИЗ
| Справочник.Версии КАК Версии");

```

```

ТекстЗапроса = СтрЗаменить(ТекстЗапроса, "&ТекстВыпущенойВерсии", НСтр("ru='выпущена'"));
ТекстЗапроса = СтрЗаменить(ТекстЗапроса, "&ТекстНевыпущенойВерсии", НСтр("ru='в разработке'"));

```

Правильно:

```

ЗапросПоВерсиям = Новый Запрос()
| ВЫБРАТЬ
| Версии.Ссылка,
| ВЫБОР КОГДА Версии.Выпущена = ИСТИНА
| | ТОГДА &ТекстВыпущенойВерсии
| | ИНАЧЕ &ТекстНевыпущенойВерсии
| КОНЕЦ КАК ТекстПояснения
| ИЗ
| Справочник.Версии КАК Версии");

```

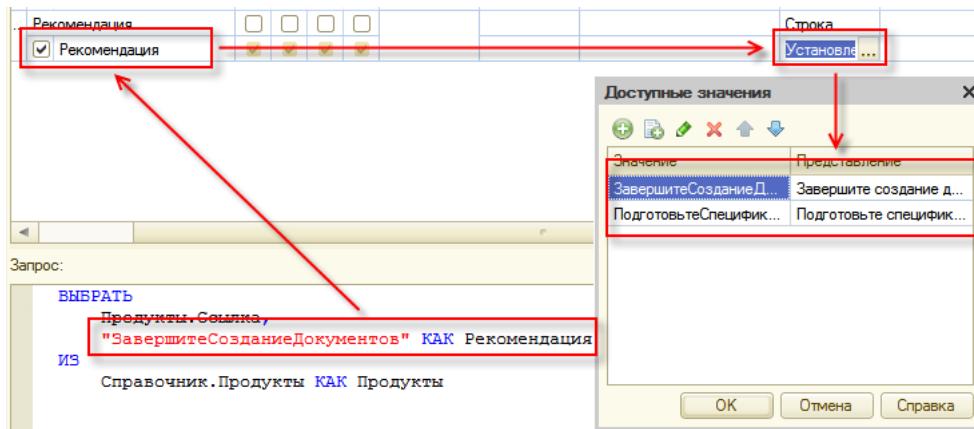
```

ЗапросПоВерсиям.УстановитьПараметр("ТекстВыпущенойВерсии", НСтр("ru='выпущена'"));
ЗапросПоВерсиям.УстановитьПараметр("ТекстНевыпущенойВерсии", НСтр("ru='в разработке'"));

```

2. Аналогичные требования предъявляются к выражениям СКД и запросам, которые используются в наборах данных СКД и содержат строковые литералы, выводимые в пользовательском интерфейсе. Например, если в выражении для параметров СКД встречаются строковые константы, требующие перевода, то следует:

а) в запросе указывать строковые константы, соответствующие "Правилам образования имен переменных" ("Завершите Создание Документов" КАК Рекомендация"), а в списке доступных значений поля указать локализуемый строковый литерал ("Завершите создание документов");



б) либо значение таких параметров с помощью функции НСтр устанавливать не в колонке Выражение, а в модуле отчета в обработчике события ПриКомпоновкеРезультата

Неправильно:

```

ВЫБОР
КОГДА ВидОперации = "Отгрузка клиентам" ТОГДА 1
КОГДА ВидОперации = "Возвраты товаров от клиентов" ТОГДА 2
КОГДА ВидОперации = "Приемка от поставщиков" ТОГДА 3
КОНЕЦ

```

Правильно:

```

ВЫБОР
КОГДА ВидОперации = &ВидОперацииОтгрузкаКлиентам ТОГДА 1
КОГДА ВидОперации = &ВидОперацииВозвратыТоваровОтКлиентов ТОГДА 2
КОГДА ВидОперации = &ВидОперацииПриемкаОтПоставщиков ТОГДА 3
КОНЕЦ

```

в) В выражениях, используемых в настройках СКД, например Выражение представления и Выражение упорядочивания на закладке Наборы данных, а также в других им подобных, необходимо использовать функцию НСтр, аналогично тому, как это делается в коде модулей (см. стандарт Интерфейсные тексты в коде: требования по локализации).

Неправильно:

```

Выбор Когда Объект = Раздел
Тогда Выбор
Когда Раздел = Значение(ПланВидовХарактеристик.РазделятьЗапретаИзменения.ПустаяСсылка)
    Тогда "<для всех разделов и объектов, кроме указанных>"
    Иначе "<для всех объектов, кроме указанных>"
Конец
Иначе Объект
Конец

```

Правильно:

```

Выбор Когда Объект = Раздел
Тогда Выбор
Когда Раздел = Значение(ПланВидовХарактеристик.РазделятьЗапретаИзменения.ПустаяСсылка)
    Тогда НСтр("&ru='<для всех разделов и объектов, кроме указанных>'")

```

```

Иначе НСтр("ги='<для всех объектов, кроме указанных>'")
Конец
Иначе Объект
Конец

```

3. Для колонок отчета на СКД для поля выборки, полученного вычислением с заданием ему псевдонима, необходимо задавать синоним при разработке. Нельзя опираться на автоматически генерированный заголовок по имени/псевдониму.

Неправильно:

Наименование	Наименование	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Description	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Цена	Цена	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Цена	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Query:

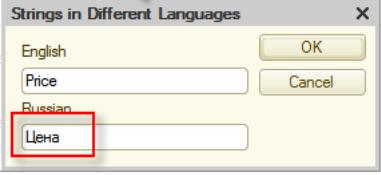
```

ВЫБРАТЬ
    Номенклатура.Наименование,
    12 КАК Цена
    ИЗ
    Справочник.Номенклатура КАК Номенклатура

```

Правильно:

Наименование	Наименование	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Description	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Цена	Цена	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Price	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



В отчётах, если не стоит галочка у поля выборки, полученного вычислением с заданием ему псевдонима, оно не попадает в результаты поиска редактирования текстов интерфейсов.

См. также

- [Оформление текстов запросов](#)

Форматирование даты, числа, Булево: требования по локализации

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

#std763

1. При использовании функции **Формат** в некоторых случаях следует использовать функцию **НСтр** при создании форматной строки (формат представления или редактирования дат, Булево значения и т.п.). При этом форматная строка, задаваемая в свойствах метаданных (форм), подлежит локализации всегда, также, как синоним.

1.1. Форматирование дат

для вывода дат следует учитывать, что в различных странах принятые различные порядок следования и разделители для составных частей даты.

Например, одна и та же дата: 20.12.2012 – для России, 12/20/2012 – для США.

Поэтому вместо явного задания формата даты рекомендуется использовать локальный формат даты (ДЛФ).

В случаях, когда использовать локальный формат даты не получается, и требуется задать произвольный формат (ДФ) или указать представление пустой даты (ДП) нужно применять функцию **НСтр** к форматной строке, чтобы при локализации оставалась возможность переопределить выводимый формат:

Неправильно:

```

Формат(ДатаУтверждения, "ДФ=дд.мм.гггг");
Формат(ДатаУтверждения, "ДФ=мммммммм гггг") + " г.";

```

Правильно:

```

Формат(ДатаУтверждения, "ДЛФ=дд");
Формат(ДатаУтверждения, НСтр("ги='ДФ='мммммммм гггг'"));

```

1.2. Форматирование числа

Следует применять функцию **НСтр** к форматной строке в случае, когда

- для числа задается нечисловое представление нулевого значения (ЧН);
- указан шаблон форматирования числа (ЧФ);
- переопределяется разделитель дробной части (ЧРД).

Неправильно:

```

Предупреждение(Формат(100, "ЧН=Отсутствует"));
Предупреждение(Формат(100, "ЧФ=\"$Ч' / Час'\""));

```

Правильно:

```

Предупреждение(Формат(100, НСтр("ги = 'ЧН=Отсутствует'")));
Предупреждение(Формат(100, НСтр("ги = 'ЧФ=\"$Ч' / Час'\""))); // "$100 / Час"

```

1.3. Форматирование Булево

Для вывода Булево значения пользователю всегда применяйте функцию **НСтр** к форматной строке.

Неправильно:

```

Предупреждение(Формат(Истина, "БЛ=Нет; БИ=Да"));

```

Правильно:

```
Предупреждение(Формат(Истина, НСтр("ru='БЛ=Нет; БИ=да'")));
```

1.4. Не следует переопределять поведение отображения локализации данных по умолчанию – формат отображения операционной системы. При использовании функции **Формат** следует избегать использовать параметр «L».

2. При задании формата в полях ввода в формах и полях отчетов на базе СКД также рекомендуется локальный формат даты. Использовать другие форматы допустимо, если по сути решаемой задачи локальный формат не подходит – тогда форматная строка будет переводиться при переводе конфигурации.

3. При переопределении стандартных представлений полей в отчетах на базе СКД следует придерживаться тех же правил, что и в коде модулей. Например, неправильно:

```
"N " + ВОтветНомер + " от " + Формат(ВОтветНадата, "ДФ=dd.ММ.ЧЧЧЧ")
```

правильное выражение, по которому вычисляется представление поля:

```
СтрШаблон(
    НСтр("ru = 'N%1 от %2'"),
    ВОтветНомер,
    Формат(ВОтветНадата, "ДФ=Д"))
```

4. В случае, когда требуется передача значения в машиночитаемом виде, вне зависимости от информационной системы и настроек локализации, применяемых в ней, вместо локализации значения следует выполнить сериализацию. Локализацию дат нужно использовать всегда, когда это возможно. В тех случаях, когда это технически нецелесообразно, допускается отказываться от локализации. Например, при генерации файла формата XML, поддерживаемого банк-клиентом системы, специфичной для России.

В общем случае для сериализации рекомендуется использовать метод **XMLСтрока**.

Для десериализации **XMLЗначение**. Или метод **ПривестиЗначение** объекта **ОписаниеТипов**.

4.1. Сериализация дат

При разработке собственных форматов передачи данных между различными системами рекомендуется сериализовать дату в формате ISO: "ГГГГ-ММ-ДДЧЧ:ММ:СС", например "2009-02-15T00:00:00Z" (соответствует типу dateTime схемы XML см. <http://www.w3.org/TR/xmlschema-2/#dateTime>).

Неправильно:

```
Строка = Формат(Дата, "ДФ=ГГГГ-ММ-ДДЧЧ:ММ:СС"); // Сериализация
```

Правильно:

```
// Сериализация
Строка = XMLСтрока(Дата); // Сериализация
// Или
Строка = ЗаписьДатуJSON(Дата, ФорматДатыJSON.ISO); // Сериализация
ОписаниеТипа = Новый ОписаниеТипов("Дата");
Дата = ОписаниеТипа.ПривестиЗначение(Строка);

// Десериализация
Дата = XMLЗначение(Тип("Дата"), Стока);
// Или
Дата = ПрочитатьДатуJSON(Стока, ФорматДатыJSON.ISO);
```

4.2. Сериализация числа

Неправильно:

```
// Сериализация
Строка = Стока(Число);
// Или
Строка = Формат(Число);
```

Правильно:

```
Строка = XMLСтрока(Число); // Сериализация
Число = XMLЗначение(Тип("Число"), Стока); // Десериализация
```

4.3. Сериализация Булево

Неправильно:

```
// Сериализация
Строка = Стока(Булево);
// Или
Строка = Формат(Булево);
// Или
Строка = Формат(Булево, "БЛ=off; БИ=on");
```

Правильно:

```
// Сериализация
Строка = XMLСтрока(Булево);
// Или
Булево = XMLЗначение(Тип("Булево "), Стока);
// Или
Строка = ?(Булево, "on", "off");
```

Строковые константные выражения в коде: требования по локализации

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

#std764

1. Не локализуются строковые константы с внутренними идентификаторами, которые не выводятся пользователем. К ним не следует применять функцию **НСтр**. При этом, такие строковые идентификаторы должны именоваться по правилам именования идентификаторов переменных встроенного языка.

Например:

```
Возврат "ОперацияВыполненаУспешно";
Оповестить("ЗаписьФайл", Новый Структура("Событие", "ВерсияСохранена"), ФайлСсылка);
```

Неправильно:

```
Если Статус = "Отгрузка клиентам" Тогда ...
```

Правильно:

Если Статус = "ОтгрузкаКлиентам" Тогда ...

Это требование распространяется и на использование идентификаторов в текстах запросов и в выражениях СКД.

При использовании в коде строковых констант можно и даже предпочтительно применять функции, возвращающие эти строковые константы, такой подход упрощает отладку и рефакторинг кода.

Правильно:

```
РезультатЗагрузки = ЗагрузитьФайлИзИнтернета(...);
Если РезультатЗагрузки = РезультатЗагрузкиУспешно() Тогда
...
Иначе Если ...
...
```

```
Функция РезультатЗагрузкиУспешно()
    Возврат "Успешно";
Конецфункции
```

Это устраниет неоднозначность, когда идентификатор визуально выглядит в коде как строка, выводимая пользователю, но без необходимого НСтр.

2. В алгоритмах программы не следует использовать представления объектов и типов. Строковые представления предназначены только для вывода пользователю, они могут различаться в зависимости от текущего языка программы. Поэтому их использование приводит к ошибкам при локализации конфигурации, а также при интеграции с системами на других языках. Частные случаи некорректного использования представлений объектов и типов в коде.

2.1. Для получения предопределенного значения на клиенте следует указывать его строковое имя, как оно указано в конфигураторе.

Например, неправильно:

```
Если Стока(ЮрФизЛицо) = "Юридическое лицо" Тогда
```

правильно

```
Если ЮрФизЛицо = ПредопределеноеЗначение("Перечисление.ЮридическоеФизическоеЛицо.ЮридическоеЛицо") Тогда
```

Подробнее см. раздел [Работа с предопределенными значениями](#) в документации к платформе 1С:Предприятие (на ИТС).

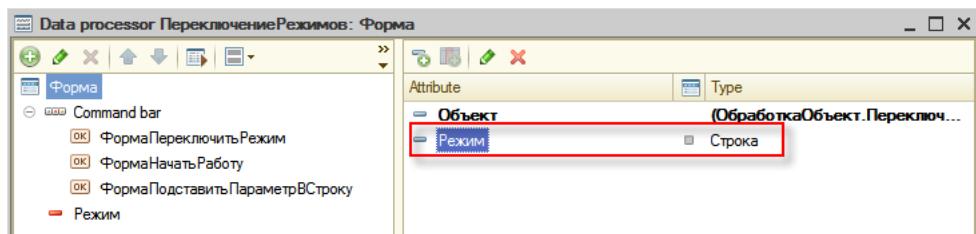
Элементы форм: требования по локализации

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

#std765

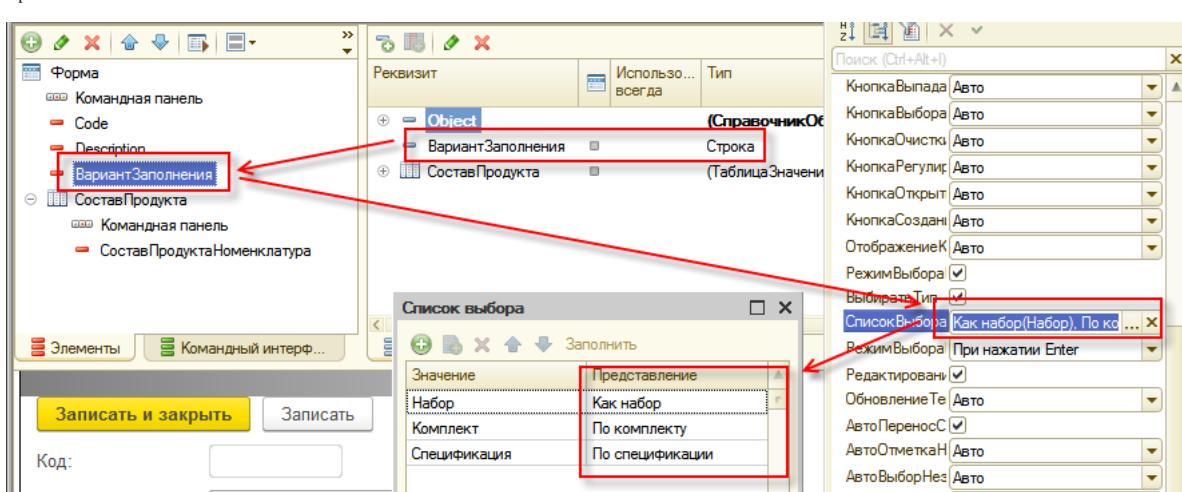
1. Нельзя присваивать реквизитам, используемым в элементах формы, строковые значения, не прошедшие локализацию. Рекомендуется для этого использовать списки значений, в которых для каждого значения задается локализуемое представление, или перечисления, если они используются в таблицах базы данных.

Неправильно:

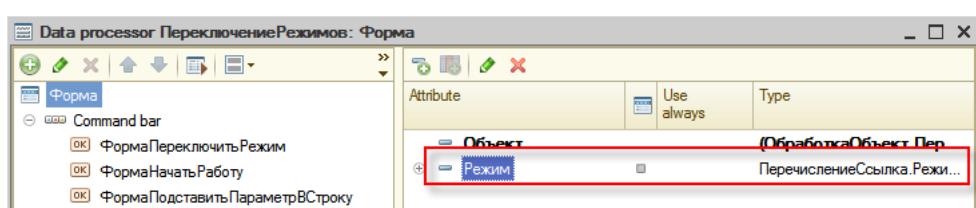


```
Если Режим = "Рабочий" Тогда
    Режим = "Демо";
Иначе
    Режим = "Рабочий";
КонецЕсли;
```

Правильно:



Правильно:



```

Если Режим = Перечисления.РежимыРаботы.Рабочий Тогда
    Режим = Перечисления.РежимыРаботы.Демо;
Иначе
    Режим = Перечисления.РежимыРаботы.Рабочий;
КонецЕсли;

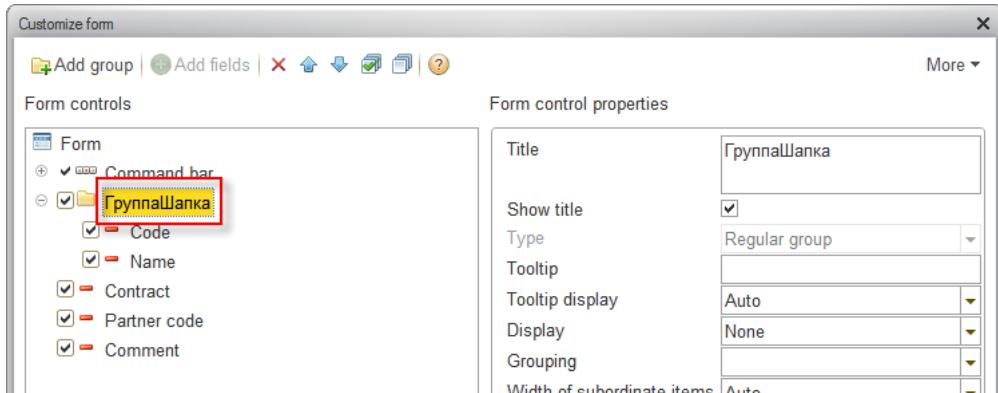
```

Методическая рекомендация (полезный совет)

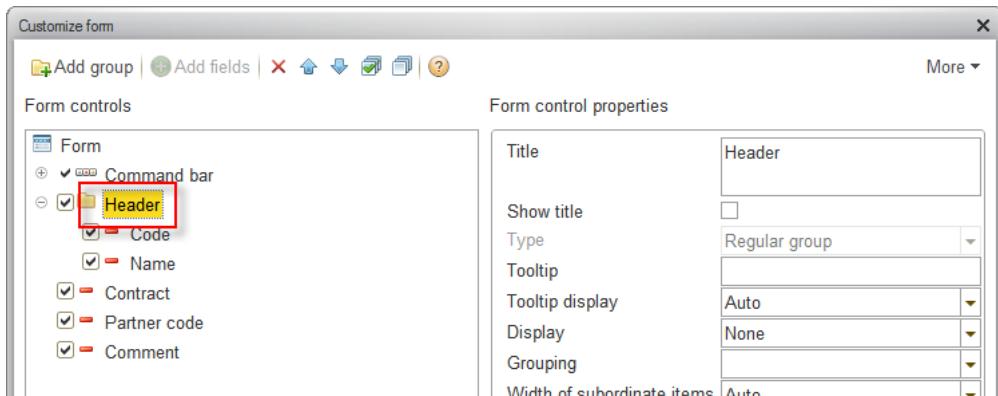
2. Для всех таблиц и групп на формах необходимо указывать заголовки, и, если эти заголовки не нужно показывать пользователю, в свойствах элемента формы в явном виде убирать отображение заголовка элемента.

Это требование связано с тем, что, если заголовок не указан, при использовании команды **Изменить форму** пользователь увидит заголовки, автоматически генерированные из имен элементов формы, а перевести их будет невозможно, потому что такие заголовки попадут в выгрузку на перевод как пустые, и переведены не будут.

Неправильно:



Правильно:



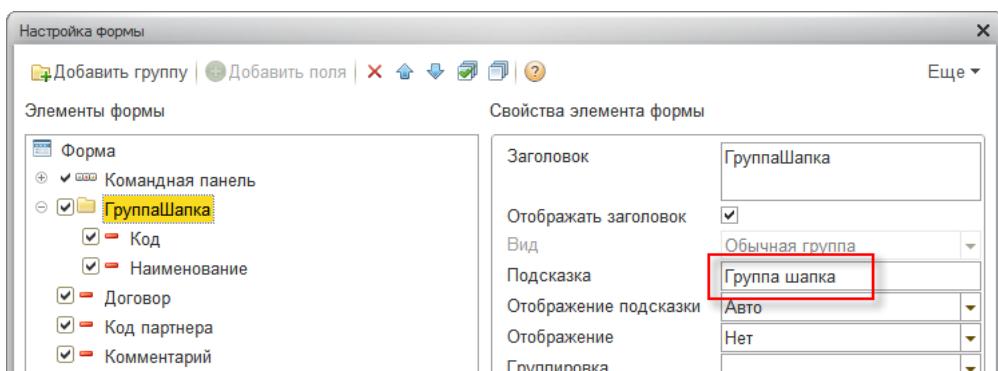
Для автоматической расстановки заголовков можно воспользоваться обработкой [автоформатирования кода и локализации](#).

3. Следует уменьшать количество незначащей информации, подлежащей локализации.

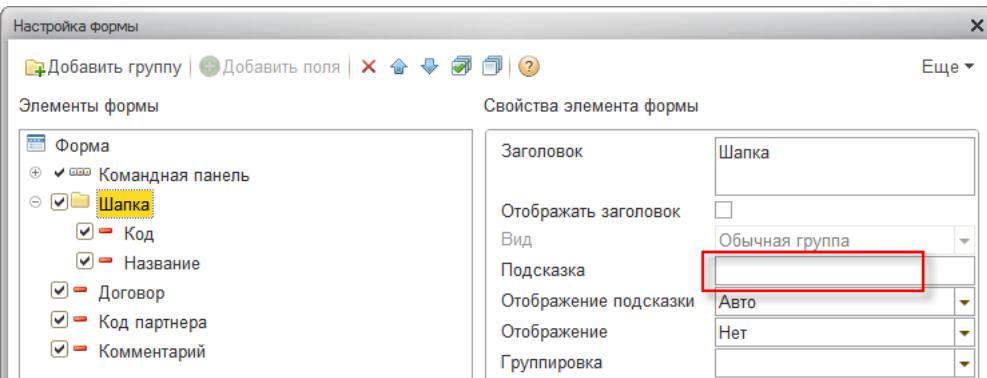
3.1. Удалять бессмысленные подсказки у групп форм. См. стандарт [Подсказка и проверка заполнения](#).

Это связано не только со стремлением удешевить перевод, но и с тем, что пользователь может увидеть такие подсказки, например, когда будет изменять форму.

Неправильно:



Правильно:



3.2. У реквизитов форм, не размещенных на форме в виде элементов управления, следует очищать заголовки, т.к. они не видны пользователю. Как правило, это служебные реквизиты, используемые для технологических целей.

Для удаления бессмысленных подсказок можно воспользоваться обработкой, приложенной к статье [Тексты модулей](#).

4. Следует задавать заголовок для колонок динамического списка, получающихся в запросе комбинацией других колонок или для которых задан свой псевдоним. Нельзя опираться на автоматически генерированный заголовок по имени/псевдониму.

Например, неправильно:

Правильно:

Примеры, когда заголовок колонок следует задавать в явном виде:

ВЫБРАТЬ
Таблица.Поле1 КАК Поле2
ВЫРАЗИТЬ(Таблица.Поле1 КАК СТРОКА(100)) КАК Поле3

В таком случае, когда поле создается в запросе и ему присваивается имя, то синоним не "подтягивается" автоматически из метаданных, т.к. не существует реквизита, связанного с этим полем. Инструментом редактирования текстов интерфейсов не находится заголовок для колонок в динамическом списке, которым задан псевдоним в запросе. Имя колонки динамического списка должно быть задано, даже если заголовок поля не выводится на форму, так как имена колонок выводятся пользователю при настройке полей формы (команда **Еще - Изменить форму...**).

5. В полях форм со списками выбора следует всегда устанавливать свойство **РежимВыбораИзСписка** в значение **Истина**. В этом случае, в поле будет корректно выводиться локализуемое представление, а не значение из списка выбора.

См. также

- [Группы элементов формы](#)

Регламентные задания: требования по локализации

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

#std767

1. Предопределенным регламентным заданиям не следует задавать наименования в конфигураторе. Вместо этого, достаточно задать синоним предопределенного регламентного задания.

Неправильно:

...: ЗагрузкаКлассификатораБанко ... X
Логотипы Помощь Поиск (Ctrl+Alt+I) X
▼Основные:
Имя ЗагрузкаКлассификатораБанков
Синоним Загрузка классификатора банков
Комментарий
Имя метода РаботаСБанками.Загрузка
Наименование Загрузка классификатора банк
Ключ
Расписание Открыть
Использование

Правильно:

...: ЗагрузкаКлассификатораБанко ... X
Логотипы Помощь Поиск (Ctrl+Alt+I) X
▼Основные:
Имя ЗагрузкаКлассификатораБанков
Синоним Загрузка классификатора банков
Комментарий
Имя метода РаботаСБанками.Загрузка
Наименование
Ключ
Расписание Открыть
Использование

Для непредопределенных (параметризованных) регламентных заданий наименование задается программно, оформляется с НСтр и может описывать контекст выполнения задания.

Правильно:

```
Задание = РегламентныеЗадания.СоздатьРегламентноеЗадание(Метаданные.РегламентныеЗадания.РассылкаОтчетов);
Задание.Наименование = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НСтр("ги = 'Рассылка отчетов: %1'", РассылкаОтчетов));
Задание.Записать();
```

Это связано с тем, что в отсутствие наименования платформа возьмет его автоматически из локализуемого синонима. Если же наименование задано, то будет использоваться оно, а оно не поддерживает локализацию.

См. также

- [Общие требования к регламентным заданиям](#)

Макеты: требования по локализации и поддержке разных языков интерфейса

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

#std766

1. Как правило, перевод табличных, текстовых и HTML-макетов на другие языки не требует каких-либо подготовительных действий при разработке конфигурации. Особым случаем являются двойочные макеты, подлежащие переводу, а также HTML-макеты, содержащие изображения, которые должны быть отдельно подготовлены на разных языках. Например, это двойочные макеты печатных форм в форматах официальных документов, HTML-макеты с инструкциями, включающие скриншоты интерфейса программы или изображения, содержащие текстовую информацию.

Такой макет необходимо пометить специальным образом, пометка будет означать, что при переводе будет необходимо создавать копию, а не переводить содержимое макета. Для этого в имени макета следует указывать постфикс в виде подчеркивания и код языка так, как он задан в метаданных, в языке **Русский** в свойстве **Код языка**: "_ru".
Например, неправильно:

макет ПФ_ODT_СчетНаОплату (макет печатной формы счета на оплату в формате OpenOffice Writer)

правильно указывать в имени постфикс основного языка:

макет ПФ_ODT_СчетНаОплату_ru

При добавлении языков интерфейса следует добавлять макеты с соответствующими постфиксами. В коде, в зависимости от языка, использовать макет с соответствующим постфиксом. Например, неправильно:

```
... = ПолучитьОбщийМакет("ПФ_ODT_СчетНаОплату");
```

правильно:

```
... = ПолучитьОбщийМакет("ПФ_ODT_СчетНаОплату" + "_" + ТекущийЯзык());
```

Кроме того, для исключения ошибок при частичном переводе конфигурации, рекомендуется выполнять получение макета в три этапа:

- Сначала по **ИмяМакета** + "_" + **ТекущийЯзык()**;
- Если не найден, то по **ИмяМакета** + "_" + **Метаданные.ОсновнойЯзык.КодЯзыка**;
- Если не найден, то по переданному имени **ИмяМакета**.
- И наконец установить свойство **КодЯзыка** (у табличного документа) или **КодЯзыкаМакета** (у текстового документа и HTML-макета), как указано далее в п.2.

2. При разработке конфигураций, рассчитанных на несколько языков интерфейса, может возникнуть задача в одном сеансе пользователя формировать печатные формы на разных языках, а не только на текущем языке интерфейса. Например, в сеансе англоязычного пользователя сформировать счет на оплату на русском языке.

Для получения данных из табличного, текстового или HTML-макета на заданном языке, отличном от языка интерфейса текущего пользователя, необходимо использовать свойство **КодЯзыка** (доступно у табличного документа) и **КодЯзыкаМакета** (у текстового документа и HTML-макета).

Правильно:

```
Макет = ПолучитьОбщийМакет("ПечатнаяФорма");
Макет.КодЯзыкаМакета = "ru";
HTMLДокумент = Макет.ПолучитьДокументHTML();
```

3. При разработке конфигураций, рассчитанных на несколько языков интерфейса, может также возникнуть задача формировать печатные формы строго на одном языке, не зависимо от текущего языка интерфейса. Примером таких макетов могут служить регламентированные формы отчетности для государственных учреждений. Например, пользователи с любым языком интерфейса должны формировать русскоязычную счет-фактуру – налоговый документ строго установленного образца в соответствии с Налоговым кодексом Российской Федерации (не существует российских счет-фактур на других языках, кроме русского).

Для табличных и HTML-макетов, которые должны выводится пользователю и на печать строго на одном языке, следует

- указывать в наименовании постфикс кода языка, аналогично п.1
- и устанавливать код языка макета аналогично п.2 при программном получении макета в коде.

Такие макеты не должны переводиться на другие языки интерфейса. При программном формировании текстов для заполнения макета следует явно указывать второй параметр в функции `НСтр()` для того, чтобы строки были сформированы на том же языке, что и макет.

Например, не правильно:

```
Макет = ПолучитьМакет("ПФ_MXL_СчетФактура");
...
Область.Текст = НСтр("ru='Заголовок печати';")
```

Правильно:

```
Макет = ПолучитьМакет("ПФ_MXL_СчетФактура_ru");
Макет.КодЯзыка = Метаданные().Языки.Русский.КодЯзыка;
...
Область.Текст = НСтр("ru='Заголовок печати';", Метаданные().Языки.Русский.КодЯзыка);
```

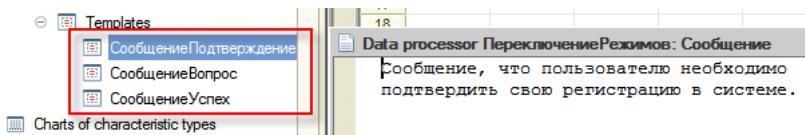
При использовании **Библиотеки стандартных подсистем (БСП)** и подсистемы **Печать** получение макета через функцию **УправлениеПечатью.МакетПечатнойФормы("ПФ_MXL_СчетФактура")** позволяет получить форму **ПФ_MXL_СчетФактура_ru** и устанавливает у макета свойство **КодЯзыка**.

4. Если в текстах макетов используются именованные параметры подстановки, необходимо соблюдать для них [требования по локализации интерфейсных текстов в коде](#).

5. Кодировку в макетах использовать UTF-8.

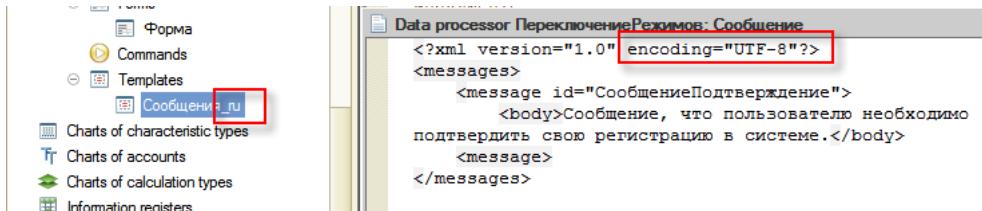
6. Также, по возможности, следует группировать однотипные макеты (использовать один макет вместо нескольких). Например, в следующем примере в конфигурации имеется несколько однотипных макетов с сообщениями, но их содержимое записывается в один справочник, поэтому правильно хранить все подобные сообщения в одном макете.

Неправильно:



```
Макет = Обработки.ПереключениеРежимов.ПолучитьМакет("Сообщение");
```

Правильно:



```
Макет = Обработки.ПереключениеРежимов.ПолучитьМакет(СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку("Сообщения_%1", ОбщегоНаз
```

7. Внешние компоненты следует размещать в макетах с типом макета **внешняя компонента**. При разработке внешней компоненты требуется обрабатывать метод **SetLocale** для локализации внешней компоненты в соответствии с полученным кодом локализации (см. [Технология создания внешних компонент](#)). Если полученный код локализации отличается от предусмотренного во внешней компоненте, то компонент должна настроить свое окружение на использование английского языка.

Денежные поля: требования по локализации

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

#std778

1. Для обеспечения работы конфигурации в странах с низким курсом национальной валюты, необходимо предусмотреть увеличение разрядности целой части числовых полей (реквизитов, ресурсов, измерений), содержащих денежный эквивалент – денежные поля. Примеры денежных полей: Сумма, Цена, Себестоимость.

Для этого в метаданных

- для денежных полей, которые могут содержать только положительные значения, вместо фиксированного типа **Число(15, 2)** использовать определяемый числовой тип **ДенежнаяСуммаНеотрицательная**;
- для денежных полей, которые могут содержать отрицательное значение, использовать тип **ДенежнаяСуммаЛибогоЗнака** со сброшенным флагом **Неотрицательное**.

В тех случаях, когда определяемый тип невозможно указать (например, в качестве типа для параметра формы или включить в составной тип), то следует задать числовой тип максимальной длины – **Число(31, 2)**. При этом в отдельных случаях эта длина должна быть снижена из-за ограничений СУБД, например, в ресурсах регистров.

2. При использовании в конфигурации **Библиотеки стандартных подсистем** не следует использовать конструктор типа **Число** для получения описания типа денежного поля. Вместо этого использовать функцию, возвращающую описание на основании определяемого типа.

Неправильно:

```
ОписаниеТиповСумма = Новый ОписаниеТипов("Число", Новый КвалификаторЧисла(15, 2));
```

Правильно:

```
ОписаниеТиповСумма = РаботаcКурсамиВалют.ОписаниеТипаДенежногоПоля();
```

3. При [использовании ВЫРАЗИТЬ в текстах запросов](#) для денежных полей, использовать приведение к типу **ЧИСЛО(31,2)**, что обеспечивает поддержку максимальной длины целой части 29. Ограничение длины целой части 29 обусловлено поддержкой сервера DB2.

Неправильно:

ВЫРАЗИТЬ(Т.Сумма / Т.Количество КАК ЧИСЛО(15,2))

Правильно:

ВЫРАЗИТЬ(Т.Сумма / Т.Количество КАК ЧИСЛО(31,2))

4. При использовании функции **Формат** и при задании формата в свойствах элементов формы, свойствах полей наборов схем компоновки данных и т.д. не следует задавать общую длину.

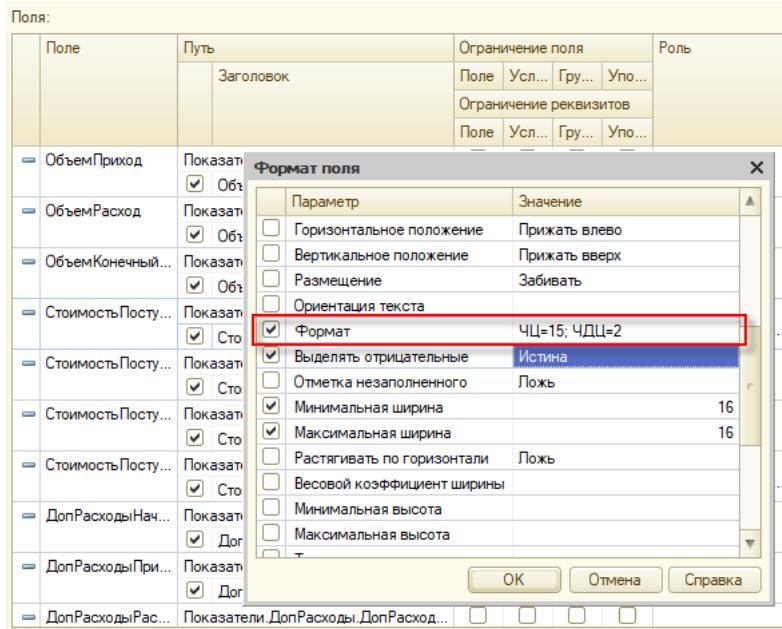
Неправильно:

Формат(Выборка.СуммаДокумента, "ЧЦ=15; ЧДЦ=2")

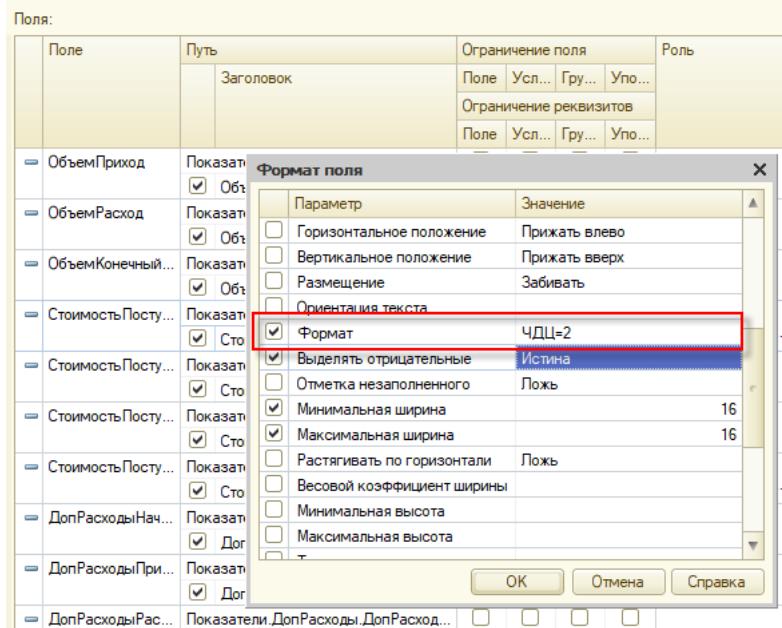
Правильно:

Формат(Выборка.СуммаДокумента, "ЧДЦ=2")

Неправильно:



Правильно:



Автогенерированные данные в информационной базе: требования по локализации

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

#std784

1. Автогенерируемые строки, которые программно записываются в информационную базу и выводятся пользователям, необходимо формировать не на языке текущего пользователя, а на языке информационной базы.

Например, при начальном заполнении информационной базы данными из макета, автогенерации комментария к проводке или определении значения параметра **ИмяСобытия** метода **ЗаписьЖурналаРегистрации**.

В противном случае, если документ провел пользователь с русскоязычным интерфейсом, а затем тот же документ перепровел пользователь с англоязычным интерфейсом, то содержание записей регистра бухгалтерии станет другим (что ошибочно).

Такие места в коде рекомендуется сопровождать комментарием, поясняющим, что строка является данными, а не интерфейсным текстом:

Неправильно:

```
Комментарий = НСтр("ru = 'Комментарий к проводке'");
```

Правильно:

```
Комментарий = НСтр("ru = 'Комментарий к проводке'", КодОсновногоЯзыка); // строка записывается в ИБ
```

Где **КодОсновногоЯзыка** – код языка хранения данных в информационной базе, выбранный в момент первого запуска программы из языков интерфейса конфигурации и сохраненный в константе **ОсновнойЯзык**.

При использовании в конфигурации **Библиотеки стандартных подсистем** для получения кода языка для хранения данных следует использовать функцию **КодОсновногоЯзыка** общего модуля **ОбщегоНазначения**.

```
Комментарий = НСтр("ru = 'Комментарий к проводке'", ОбщегоНазначения.КодОсновногоЯзыка()); // строка записывается в ИБ
```

2. Аналогичные требования распространяются и на обработчики начального заполнения заполняющих строковые реквизиты предопределенных элементов справочников, ПВХ и т.п.

Правильно:

```
Процедура ПриДобавленииОбработчиковОбновления(Обработчики) Экспорт
```

```
Обработчик = Обработчики.Добавить();
```

```
Обработчик.НачальноеЗаполнение = Истина;
```

```
Обработчик.Процедура = "ПользователиСлужебный.ЗаполнитьНаименованиеПредопределенныхГруппПользователей";
```

```
КонецПроцедуры
```

```
Процедура ЗаполнитьНаименованиеПредопределенныхГруппПользователей() Экспорт
```

```
ГруппаПользователей = Справочники.ГруппыПользователей.ВсеПользователи.ПолучитьОбъект();
```

```
ГруппаПользователей.Наименование = НСтр("ru='Все пользователи'", КодОсновногоЯзыка);
```

```
ОбновлениеИнформационнойБазы.ЗаписатьОбъект(ГруппаПользователей);
```

```
КонецПроцедуры
```

При добавлении новых предопределенных элементов, необходимо создать новый обработчик начального заполнения с указанием версии или дополнить существующий. См. также стандарт [Обработчики обновления информационной базы](#).

При использовании в конфигурации **Библиотеки стандартных подсистем** не следует создавать собственные обработчики начального заполнения. Первоначальные данные заполнения нужно размещать в модуле менеджера объекта в процедуре **ПриНачальномЗаполненииЭлементов**. Тогда для объектов, зарегистрированных в процедуре **ПриОпределенииНастроек** общего модуля **ОбновлениеИнформационнойБазыПереопределяемый**, эти процедуры будут вызваны автоматически.

При добавлении нового или изменении существующего элемента следует создать собственный обработчик обновления перехода на версию и продублировать изменения в процедуре **ПриНачальномЗаполненииЭлементов** модуля менеджера объекта. Подробнее см. [документацию к БСП](#).

Размеры экрана

Область применения: управляемое приложение.

#std727

1. При проектировании интерфейсов типовым разрешением экрана следует считать 1280x768 пикселей. Разработку (конфигурирование) нужно вести в стандартном разрешении - 96 DPI.

2. Условия эксплуатации:

- Основное окно программы растянуто на весь экран
- Панель задач операционной системы видна, расположена в нижней части экрана и уменьшает его высоту на 40 пикселей
- Программа открыта в браузере. Высота рабочей области сокращается из-за заголовков и панелей браузера в среднем на 60 пикселей (Internet Explorer – на 54, Mozilla Firefox – на 63, Google Chrome – на 60).

3. Исходя из вышеприведенных значений, область, доступная для работы с программой, имеет размер 1280x668 пикселей.

4. При типовом разрешении экрана элементы внутри форм должны помещаться без вертикальной и горизонтальной полос прокрутки. Исключение составляют списки, в которых вертикальная полоса прокрутки допускается, а горизонтальная – является нежелательной.

См. также: [Общие рекомендации \(8.2\)](#).

Оформление групп разделов с настройками и справочниками

Область применения: управляемое приложение.

#std753

1. Группа раздела может включать в себя:

- Заголовок (1)
- Реквизиты и настройки (2)
- Информационные сообщения (3).

1.1. Типовым разрешением экрана считается 1280x768 px. Все размеры в стандартах приведены с учетом этих значений.

Например:



Продажи

Управление параметрами отражения операций оптовых и розничных продаж. Учет заказов клиентов, документов продаж, счетов на оплату, сделок с клиентами, комиссионных продаж, возвратов от клиентов и параметров согласования заявок и документов продаж.

Оптовые продажи

1

Использование соглашений с клиентами:

Типовые и индивидуальные соглашения

Типовые соглашения - это типовые условия продаж, которые используются при продажах товаров. В каждом типовом соглашении указываются условия продаж: виды цен, график оплат и т.д.

Индивидуальные соглашения заключаются с клиентами в том случае, если им предоставляются особые условия, которые отличаются от типовых условий продаж. Индивидуальные соглашения могут являться уточнениями к типовым.

Договоры с клиентами

Регистрация договоров с клиентами, возможность отражения расчетов в разрезе договоров.

Настройки продавцов

Настройка ограничений ручных скидок (наценок) по пользователям и группам пользователей, настройка прав РМК.

Заказы клиентов

Заявки на возврат

Учет и управление запросами клиентов на поставку товаров или оказание услуг, контроль оплаты и отгрузки.

Возврат товаров от клиента, их замена, возврат денежных средств клиенту.

Невозможно отключение заказов клиентов, так как включено ведение работы через торговых представителей.

3

2. Разделы, состоящие из нескольких групп, должны быть реализованы в виде сворачиваемых групп. Для правильного отображения заголовка сворачиваемой группы в Конфигураторе используются следующие параметры:

- Поведение: Свертываемая
- Отображение: Картинка
- Отображение: Обычное выделение
- ЦветТекстаЗаголовка: Авто
- ШрифтЗаголовка: Авто.

Например:

[Заказы поставщикам](#)

[Документы закупок](#)

[Импорт](#)

2.1. Рекомендуется формировать группы раздела таким образом, чтобы при типовом разрешении 1280x768 пикселей все группы умещались на экране. Следует сворачивать группы, по возможности избегая появления вертикальной прокрутки.

Например, в разделе "Продажи" все группы свернуты по-умолчанию:

Демонстрационная база / Соколов Максим Игоревич / 1С:ERP Управление предприятия... (1С:Предприятие)

Продажи

Управление параметрами отражения операций оптовых и розничных продаж. Учет заказов клиентов, документов продаж, счетов на оплату, сделок с клиентами, комиссионных продаж, возвратов от клиентов и параметров согласования заявок и документов продаж.

?

- Оптовые продажи
- Розничные продажи
- Торговые представители
- Производство из давальческого сырья
- Настройки использования мобильного приложения "1С:Заказы"

2.2. Допускается оставлять первую группу раздела развернутой, если это не повлечет за собой "растягивания" формы за пределы типового разрешения экрана 1280x768 пикселей.

Например, в разделе "Закупки" первая группа развернута — остальные группы в свернутом виде не выходят за рамки экрана:

Демонстрационная база / Соколов Максим Игоревич / 1С-ERP Управление предприятия... (1С:Предприятие)

Закупки

Управление параметрами отражения операций закупок, таких как заказы поставщикам, поступления товаров и услуг, комиссионные закупки, заказы на внутреннее потребление, перемещение и сборку (разборку).

Соглашения и договоры с поставщиками

Соглашения с поставщиками

Соглашения — это условия закупок, которые используются при закупках товаров. В каждом соглашении указываются условия закупок: виды цен, график оплаты и т.д.

Договоры с поставщиками

Регистрация договоров с поставщиками, возможность отражения расчетов в разрезе договоров.

Заказы поставщикам

Документы закупок

Импорт

Статусы документов

3. Раздел, состоящий из одной группы, должен быть развернутым и иметь параметры "Поведение: Обычное". Заголовок в такой группе не используется.

Например:

Демонстрационная база / Соколов Максим Игоревич / 1С:ERP Управление предприятия... (1С:Предприятие)

Международный финансовый учет

Управление параметрами международного финансового учета.

Международный финансовый учет

Подготовка отчетности в соответствии с принципами МСФО или иными правилами, построенными на основе МСФО.

Проводки по данным оперативного учета

Формировать проводки международного учета отражением хозяйственных операций оперативного учета.

Функциональная валюта: RUB

Валюта, используемая в основной экономической среде, в которой предприятие осуществляет свою деятельность. Функциональная валюта международного финансового учета — это валюта, в которой предприятие осуществляет большую часть своих операций.

Проводки по данным регламентированного учета

Формировать проводки международного учета трансляцией проводок регл. учета.

Валюта представления: EUR

Валюта, в которой представляется международная финансовая отчетность.

Валюты

Добавление, редактирование, удаление валют, используемых при расчетах на предприятии.

Учет внеоборотных активов

Использование отдельных документов ведения основных средств и нематериальных активов для международного учета.

Загрузить модель учета

Загрузка из файла методической модели учета: плана счетов, шаблонов проводок, финансовых отчетов, групп финансового учета, статей ДДС, доходов, расходов, активов и пассивов.

Выгрузить модель учета

Выгрузка в файл методической модели учета: плана счетов, шаблонов проводок, финансовых отчетов, групп финансового учета, статей ДДС, доходов, расходов, активов и пассивов.

Общие принципы построения командного интерфейса

Область применения: управляемое приложение.

#std711

Командный интерфейс – средство навигации пользователей по функциональности конфигурации. Он включает в себя:

- Панель разделов
- Панель функций текущего раздела
- Меню функций
- Команды навигации и действий

Каждый из этих элементов имеет свое назначение, но все вместе они создают пространство команд – возможностей для пользователя.

Состав и расположение панелей можно настраивать. При этом следует руководствоваться спецификой прикладного решения, а также следующими принципами:

1. Количество элементов в панелях и меню должно быть таким, чтобы при стандартном разрешении экрана они помещались без прокрутки
2. Элементы внутри панелей и меню следует располагать в порядке убывания важности и частоты использования. Наиболее приоритетные пункты и команды следует располагать первыми.
3. Не рекомендуется располагать рядом команды и пункты меню, в названиях которых совпадают первые символы.

Правильно

Регламентированные отчеты
Отчеты для руководителя

Неправильно

Регламентированные отчеты
Регламентные операции

4. Командный интерфейс нужно проектировать таким образом, чтобы он способствовал повышению эффективности выполнения повседневной работы и быстрому освоению программы. Чтобы этого добиться, при разработке следует учитывать мнение пользователей и их представление о том, как команды должны быть сгруппированы.

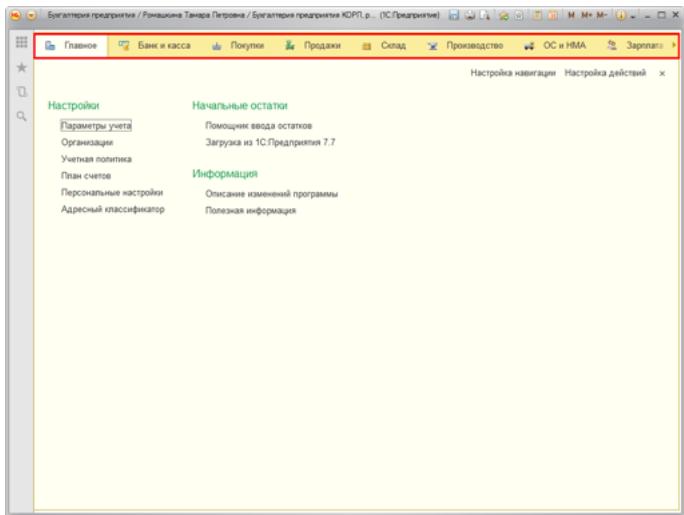
Командный интерфейс (8.2)

Панель разделов

Область применения: управляемое приложение.

#std712

Панель разделов может выводиться как в виде отдельной панели, так и в составе меню функций:



1. Состав панели

Панель разделов является "лицом" программы и имеет большое значение на этапе освоения, поэтому состав и порядок разделов следует проектировать с особой тщательностью. Количество и состав разделов должен соответствовать реальным участкам работ и областям деятельности так, как их видят пользователи.

1.1. Раздел "Главное" по умолчанию присутствует во всех конфигурациях и располагается первым. В него рекомендуется добавлять команды перехода к объектам, относящимся ко всей конфигурации, и не включать объекты, относящиеся к конкретным разделам учета или областям деятельности.

1.2. Списки с условно-постоянной информацией (справочники, регистры сведений, списки перечислений и другие), в командном интерфейсе можно размещать:

- внутри раздела, с которым связан такой список (в конце меню или в "См. также")
[Например, список должностей в разделе Кадры](#)
- в специально выделенном разделе конфигурации. При этом в других разделах команды перехода к подобным спискам, как правило, не размещаются
[Например, разделы "Справочники", "Настройки", "НСИ".](#)
- в панели навигации формы списка с условно-постоянной информацией, которой они "подчинены"
[Например, "Счета учета номенклатуры" в панели навигации справочника "Номенклатура"](#)

Иключение составляют списки с условно-постоянной информацией, с которых начинаются бизнес-процессы. Такие списки рекомендуется помещать в группы меню соответствующего раздела.

[Например, справочник "Сотрудники" размещается в разделе "Кадры", потому что сначала в него вводится информация о работнике, а потом на основании этих данных создается приказ о приеме на работу и другие документы.](#)

1.3. Разделы для настройки, администрирования и выполнения сервисных действий следует располагать в конце панели.

2. Названия разделов

2.1. Общая длина названия раздела не должна превышать 35 знаков с учетом пробелов (это позволяет разместить название в 2 строки, при дальнейшем увеличении количества знаков появится многоточие). Лучше выбирать названия примерно одного размера по ширине, чтобы они смотрелись единообразно и ровно.

2.2. Страйтесь сделать названия разделов конкретными и запоминающимися. Назначение раздела должно быть понятно из его названия.

2.3. По возможности не используйте длинные слова. При выборе названия рекомендуется комбинировать слова следующим образом:

Комбинация слов	Пример
Одно-два средних слова	Продажи
Одно длинное и одно короткое	Зарплата и персонал
Два коротких и одно длинное	Отчетность и справки

2.4. Используйте в названиях только общеупотребительные и соответствующие целевой аудитории сокращения и аббревиатуры.
[Например, "НДС" или "МСФО".](#)

При этом сокращение обязательно нужно расшифровать во всплывающей подсказке.

[Например, для названия раздела "ОС и НМА" появляется всплывающая подсказка "Основные средства и нематериальные активы"](#)

2.5. Не рекомендуется делать раздел с названием "Сервис", т.к. он будет перекликаться с пунктом "Сервис" главного меню и группой "Сервис" в панели / меню функций

3. Картинки разделов

3.1. Названия разделов рекомендуется выводить в режиме "Картишка и текст"

3.2. Картинки следует делать разными по начертанию и ведущим цветам для лучшей запоминаемости. Но при этом они должны быть нарисованы в одном стиле, с одинаковым направлением света. Картинки должны быть нарисованы во фронтальной плоскости проекции

См. также:

[Панель разделов \(8.2\)](#)

[Названия разделов \(8.2\)](#)

[Картинки разделов \(8.2\)](#)

[Подсказки для интерфейсных подсистем \(8.2\)](#)

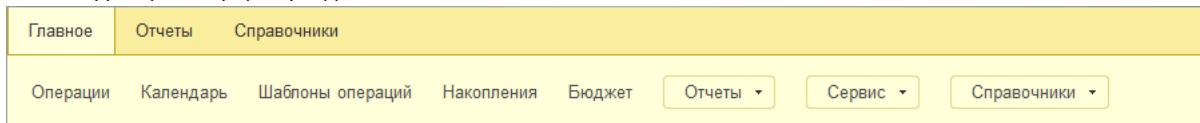
Навигация внутри раздела

Область применения: управляемое приложение.

#std714

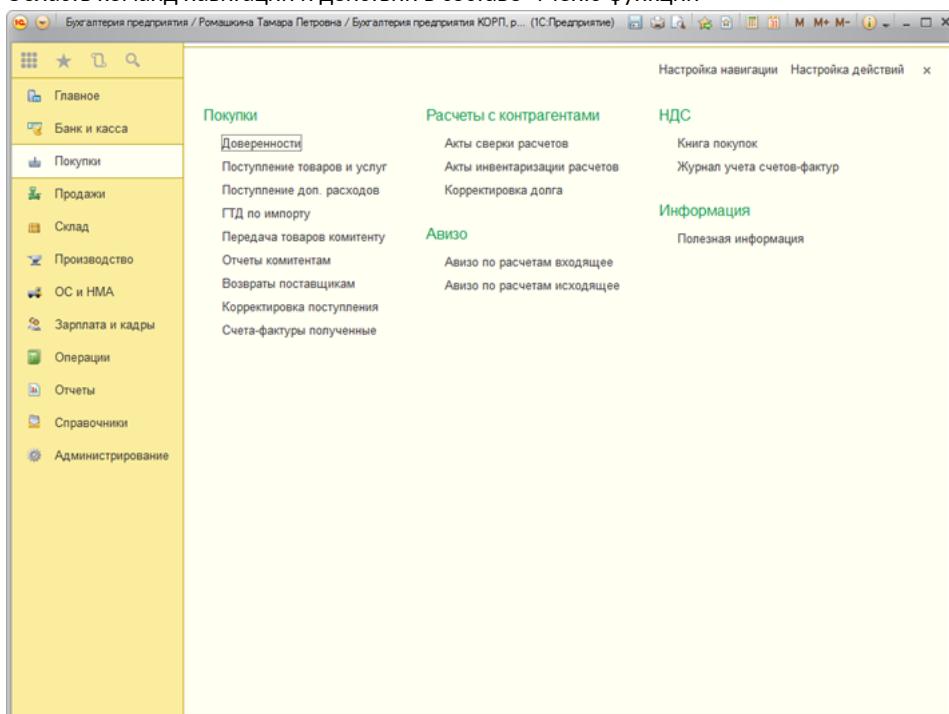
Навигация внутри конкретного раздела осуществляется с помощью "Панели функций текущего раздела" или области навигации и действий в составе "Меню функций"

Панель функций текущего раздела



Панель функций текущего раздела рекомендуется использовать в конфигурациях, содержащих небольшое количество команд в каждом разделе. В конфигурациях со сложной структурой и большим количеством команд эту панель рекомендуется скрывать.

Область команд навигации и действий в составе "Меню функций"



Область команд навигации и действий показывает все функциональные возможности раздела. В отличие от панели функций текущего раздела, команды здесь можно группировать.

1. Состав команд

1.1. В раздел рекомендуется помещать команды перехода к:

- спискам и журналам документов
- спискам "первичных" документов (данных), с которых начинаются бизнес-процессы
- рабочим местам
- специальным обработкам, похожим на обычные списки
- отчетам

При этом все эти команды должны решать задачу по конкретному участку работ или области деятельности.

1.2. Второстепенные и подчиненные другим объекты можно не выносить в командный интерфейс.

Чаще всего это справочники и регистры сведений, доступ к которым можно получить из документов или других объектов.

Например:

Справочники "Контрагенты", "Договоры" выносятся в командный интерфейс:

- эти справочники открываются из разных документов, но они могут использоваться и без привязки к документам, например, для просмотра контактной

Справочник "Причины списания основного средства" не выносится в командный интерфейс:
- этот справочник можно открыть из одного специализированного документа "Списание основного средства"

информации по контрагенту или условий оплаты по договору
- справочники могут содержать много элементов, может потребоваться работа с их списком. Например, поиск договоров по статусу или периоду, удаление дубликатов контрагентов

- элементов этого справочника мало и они редко изменяются, у пользователя нет необходимости просматривать список этих элементов в отрыве от контекста

2. Названия команд

2.1. Чтобы при стандартном разрешении экрана не возникало полос прокрутки, рекомендуется следить, чтобы названия команд не превышали 38 символов, а лучше – умещались в 30.

3. Группировка команд

3.1. Область команд включает в себя блок с командами навигации и блок с командами действий, но для пользователя все они выглядят одинаково. Поэтому при группировке следует объединять команды не по техническим признакам (вид объекта; действие, происходящее при выборе команды), а по тому, как их структурирует пользователь в своей работе.

3.2. В одну группу рекомендуется включать не более 5-6 команд

3.3. Не рекомендуется делать группы, содержащие всего одну команду.

Правильно

Удержания

Алименты и другие удержания
Исполнительные листы

Неправильно

Удержания

Алименты и другие удержания

Запрещается делать группы, в которых выводится только одна "важная" команда.

Правильно

Квартальная отчетность в ПФР
Пачки, реестры, описи кварт. отчетности
Пачки АДВ-1
Пачки АДВ-2
Пачки АДВ-3
Пачки ДСВ-1
Пачки СПВ-1

или

Квартальная отчетность в ПФР
Пачки, реестры, описи кварт. отчетности

Пачки АДВ-1
Пачки АДВ-2
Пачки АДВ-3
Пачки ДСВ-1
Пачки СПВ-1

Неправильно

Квартальная отчетность в ПФР
Пачки, реестры, описи кварт. отчетности
Пачки АДВ-1
Пачки АДВ-2
Пачки АДВ-3
Пачки ДСВ-1
Пачки СПВ-1
Реестры ДСВ-3
Сотрудники
Все отчеты персчета

3.4. Группам рекомендуется добавлять название. При этом оно не должно пересекаться с названиями системных групп ("Создать", "Отчеты", "Сервис", "Важное", "См. также"). Обязательно нужно добавлять заголовок для групп, состоящих из нескольких команд и содержащих только одну важную.

Правильно

Планирование

Заказы на ремонт
Ремонтные мероприятия

Неправильно

Заказы на ремонт

Ремонтные мероприятия

См. также:

[Панель навигации основного окна \(8.2\)](#)

[Порядок и названия команд в ПН \(8.2\)](#)

[Группа команд «Важное» в ПН \(8.2\)](#)

[Группировка команд в ПН \(8.2\)](#)

[Группа «См. также» в ПН \(8.2\)](#)

[Команды, размещаемые в ПН \(8.2\)](#)

[Панель действий \(8.2\)](#)

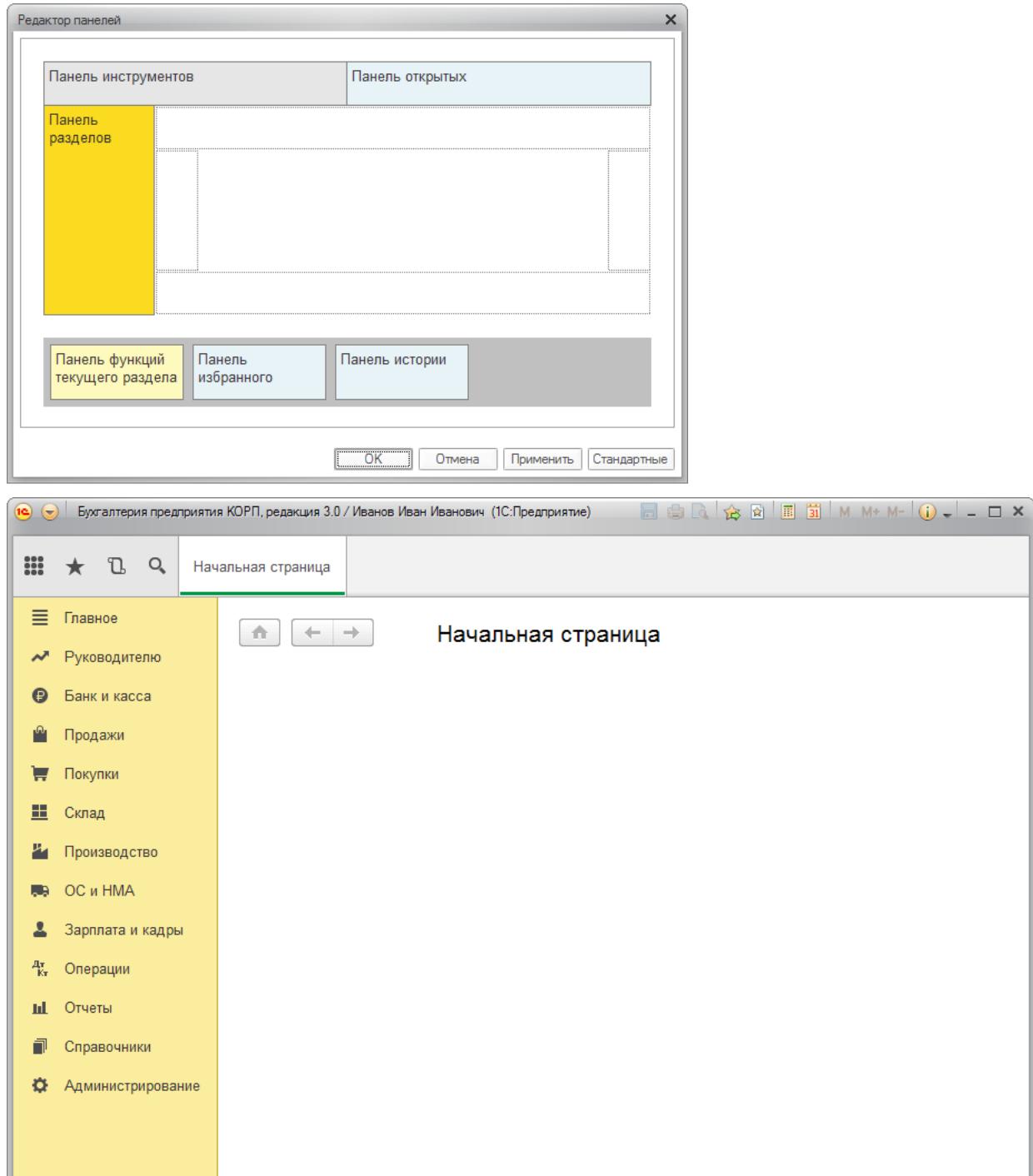
Как вместить большое количество команд

Область применения: управляемое приложение.

#std715

В конфигурациях со сложной структурой меню и большим количеством команд рекомендуется:

1. Панель разделов размещать вертикально, в левой части окна
2. Заголовки разделов выводить в виде "Картинка и текст", использовать картинки размером 16x16
3. Панель инструментов и панель открытых выводить сверху
4. Панель функций текущего раздела не отображать. Навигация внутри раздела будет осуществляться с помощью Меню функций



Компоновка форм

Область применения: управляемое приложение.

#std722

Содержание:

1. Общие принципы
2. Вкладки
3. Вспомогательные формы
4. Свертываемые группы

1. Общие принципы

1.1. Внутри одной формы все заголовки для однострочных полей ввода, переключателей, тумблеров следует располагать одинаково – или слева, или сверху.

Правильно

Налог на прибыль	НДС	ЕНВД	Запасы	Затраты на производство	Резервы
Способ оценки материально-производственных запасов (МПЗ):					
По средней стоимости					
Способ оценки товаров в рознице:					
По стоимости приобретения					

Неправильно

Налог на прибыль	НДС	ЕНВД	Запасы	Затраты на производство	
Способ оценки материально-производственных запасов (МПЗ):					
По средней стоимости					
Способ оценки товаров в рознице:					
По стоимости приобретения					

Налог на прибыль	НДС	ЕНВД	Запасы	Затраты на производство	Резервы
Способ оценки материально-производственных запасов (МПЗ):					
По средней стоимости					
Способ оценки товаров в рознице:					
По стоимости приобретения					

Заголовки многостраничных полей рекомендуется располагать в зависимости от общей композиции формы.

1.1.1. Если в группе однострочных полей заголовок одного поля существенно длиннее остальных заголовков, рекомендуется сократить название заголовка этого поля или переносить часть такого заголовка на другие строки.

Правильно

Операция:	Перемещение товаров
Перемещение под деятельность:	

Неправильно

Операция:	Перемещение товаров
Перемещение под деятельность:	

1.1.2. Поля взаимосвязанных реквизитов, находящиеся в одной группе, должны выравниваться по опорной линии.

Правильно

Статья расходов:	Прочие убытки
Аналитика расходов:	Бухгалтерия
Руководитель:	Дубинин П.Н.
Главный бухгалтер:	Батурина О.Н.

Неправильно

Статья расходов:	Прочие убытки
Аналитика расходов:	Бухгалтерия
Руководитель:	Дубинин П.Н.
Главный бухгалтер:	Батурина О.Н.

1.2. Элементы, которые в разных формах выполняют одну и ту же функцию, следует называть одинаково и всегда выводить примерно в одном и том же месте.

Например, реквизит "Контрагент" располагается в левой колонке шапки во всех документах, где выводится.

1.3. Если один из элементов интерфейса влияет на другие элементы, то он должен быть расположен до этих элементов (выше или левее, в зависимости от ситуации). При работе с формой пользователь сначала должен выполнить действия, связанные с основным элементом, а потом уже перейти к работе с зависимыми.

Например, выбор системы налогообложения влияет на состав вкладок, поэтому тумблер для его выбора расположен выше всех вкладок, а не на одной из них:

Система налогообложения:	Общая	Упрощенная		
УСН	ЕНВД	Запасы	Затраты на производство	Резервы

Например, поле "Договор" заполняется после того, как выбран "Контрагент", поэтому расположено в колонке справа от него. Пользователь сначала укажет контрагента в левой колонке, а потом – договор в правой:

Поставщик:	Группа Альфа	Организация:	Торговый дом "Комплексный"
Контрагент:	Альфа-1	Договор:	Комиссионный договор
Соглашение:	Поступление на комиссию	Склад:	Склад Тамбов (оптовый)

Аналогично, если внутри формы есть взаимосвязанные табличные части, то они должны располагаться рядом друг с другом.

Например, в документе "Отчет комиссионера (агента) о продажах" состав товаров и услуг зависит от выбранного покупателя. Поэтому эти табличные части расположены рядом:

Отчет комиссионера (агента) о продажах (создание) *

Провести и закрыть Записать Провести Печать Еще ?

N	Покупатель	Всего	НДС	СФ	Дата СФ	Счет-фактура
1	Никитина И.В.	2 180,00	180,00	<input type="checkbox"/>		
2	Вега-транс			<input type="checkbox"/>		

Tовары Услуги
Добавить Подбор Еще
N Номенклатура Количество Цена Сумма Цена передачи Сумма передачи
1 Минеральное масло 10,000 100,00 1 000,00
2 Сульфат аммония 5,000 200,00 1 000,00

Комментарий: Ответственный: Любимов Валерий Юрьевич

Обязательно к соблюдению

1.3.1. Использование вертикальной зеленой черты для выделенных отдельных элементов и групп взаимосвязанных реквизитов не допускается. В качестве выделения следует использовать состояния:

- слабое выделение;
- обычное выделение;
- нет.

Правильно

Комплект

Номенклатура:	<input type="text"/>
Характеристика:	<input type="text"/>
Назначение:	<для товаров>
Комплектация:	<input type="text"/>

Планирование

Желаемая дата поступления:	<input type="text"/> ..
Длительность сборки/разборки:	0 дней

Неправильно

Комплект

Номенклатура:	<input type="text"/>
Характеристика:	<input type="text"/>
Назначение:	<для товаров>
Комплектация:	<input type="text"/>

Планирование

Желаемая дата поступления:	<input type="text"/> ..
Длительность сборки/разборки:	0 дней

Правильно

Основное Товары (3) Доставка Дополнительно

Реализация по заказу Заказ клиента 00ЦУ-000008 от 15.08.2013 16:22:37

Неправильно

Основное Товары (3) Доставка Дополнительно

Реализация по заказу Заказ клиента 00ЦУ-000008 от 15.08.2013 16:22:37

Использование сильного выделения с вертикальной зеленой полосой допускается в тех случаях, когда группы взаимосвязанных реквизитов располагаются в несколько колонок по горизонтали.

Например, в документе "Шаблон магнитных карт" соблюдается четкое визуальное разделение групп реквизитов:

1-я дорожка

Используется

Предфикс шаблона: % Длина кода: 79

Суффикс шаблона: ? Разделитель блоков: =

Добавить **Еще ▾**

N	Поле	Номер первого символа
	Номер блока	Длина поля

2-я дорожка

Используется

Предфикс шаблона: ; Длина кода: 40

Суффикс шаблона: ? Разделитель блоков: =

Добавить **Еще ▾**

N	Поле	Номер первого символа
	Номер блока	Длина поля

3-я дорожка

Используется

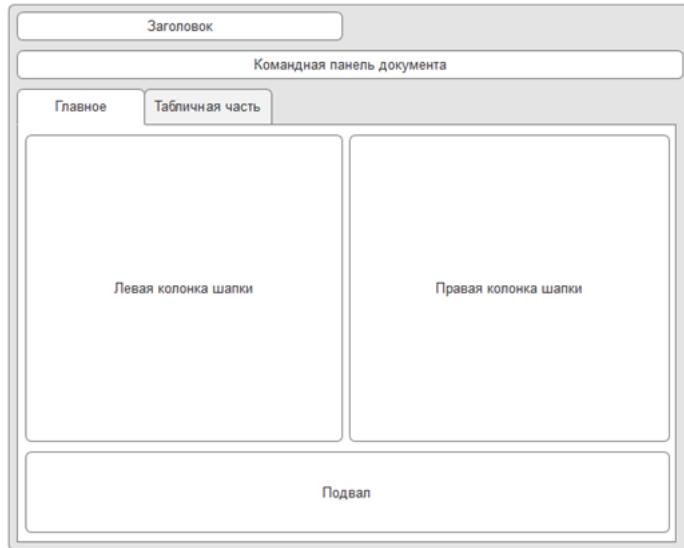
Предфикс шаблона: ; Длина кода: 107

Суффикс шаблона: ? Разделитель блоков: =

Добавить **Еще ▾**

N	Поле	Номер первого символа
	Номер блока	Длина поля

1.4. Внутри формы документа рекомендуется группировать элементы следующим образом:



В левой и правой колонках шапки следует располагать реквизиты, необходимые для правильного отражения документа в программе, при этом следует распределять реквизиты по следующим признакам:

Левая колонка	Правая колонка
- важные - заполняются пользователем вручную - обязательны для заполнения - заполняются на основе входящих данных, полученных извне	- вспомогательные, малозначительные - чаще всего заполняются автоматически - не обязательны для заполнения - значения реквизитов устанавливаются пользователем самостоятельно, по внутренним данным
Например: - Номер и дата - Контрагент - Договор	Например: - Организация - Подразделение

В нижней части формы располагается область с итогами и подвал. В подвале могут располагаться поля "Ответственный" и "Комментарий", ссылка на счет-фактуру и другие.

См. также:

- [Итоги в документах](#)
- [Поля "Ответственный" и "Комментарий"](#)

1.4.1. При компоновке документов, содержащих табличную часть, следует ориентироваться на видимое количество строк в табличной части. Табличная часть документа должна содержать не менее 7-ми строк (оптимально — не менее 10-ти строк). Если количество строк в табличной части менее 7-ми — рекомендуется размещать табличную часть на отдельной вкладке.

Например, в документе "Отчет комитету" реквизиты шапки размещены во вкладке с основными реквизитами:

Отчет комитенту АН99-000001 от 17.07.2015 (Отчет о продажах)

Провести и закрыть **Записать** **Провести** **Дт
Кт** **Печать** **Создать на основании** **Еще** **?**

Главное Товары и услуги (2) Денежные средства Расчеты

Номер: АН99-000001 от: 17.07.2015 15:40:40 Организация: Андромеда ООО

Контрагент: ОАО МГТС [НДС в сумме](#)

Договор: 10 от 10.01.2015

Комиссионное вознаграждение

Способ расчета: Процент от суммы продажи Услуга по вознаграждению:

Вознаграждение: 10,00 Счет учета доходов:

Счет учета НДС: Субконто:

Ставка НДС: Без НДС

Счет-фактура на сумму вознаграждения: Выписать счет-фактуру Всего: 5 000,00 руб. НДС (в т.ч.): 0,00

Всего: 50 000,00 руб.

Комментарий: Ответственный:

Таким образом, количество строк таблицы вкладки "Товары" реквизитами не ограничивается:

Отчет комитенту АН99-000001 от 17.07.2015 (Отчет о продажах)

Провести и закрыть **Записать** **Провести** **Дт
Кт** **Печать** **Создать на основании** **Еще** **?**

Главное Товары и услуги (2) Денежные средства Расчеты

Добавить Заполнить Подбор Еще

N	Номенклатура	Количество	Цена поступления	Сумма поступления	Цена	Сумма	Вознаграждение с НДС
1	Минеральное масло	100,000	1 000,00	100 000,00	100,00	10 000,00	1 000,00
2	Минеральные удобрения	200,000	1 000,00	200 000,00	200,00	40 000,00	4 000,00

Комментарий: Ответственный:

1.4.2. Если на форме имеются несколько невзаимосвязанных табличных частей, то их следует размещать на отдельных вкладках.

1.5. Формы следует проектировать таким образом, чтобы они не были перегружены функциями, реквизитами и элементами. Чтобы этого достичь, можно использовать:

- возможность скрытия элементов в зависимости от настроек программы или функциональных опций
 - размещение части элементов на отдельных вкладках, в свертываемых группах или во вспомогательных формах
- Конкретный способ выбирается в зависимости от ситуации и решаемых задач.

2. Вкладки

2.1. Название вкладки должно отражать суть того, что на ней располагается

Например, по названию вкладки "Товары" понятно, что на ней отображается список товаров:

Товары (1) Услуги							
Добавить		Заполнить		Подбор		Изменить	
N	Номенклатура	Количество	Цена	Сумма	% НДС	НДС	Всего
1	Комплект постельного белья	1,000	1 000,00	1 000,00	Без НДС		1 000,00

2.2. Поля, обязательные для заполнения, следует располагать на вкладке, активной в момент открытия формы

2.3. Не рекомендуется делать вкладки, содержащие небольшое число элементов (1-2), если они не занимают все пространство вкладки

3. Вспомогательные формы

3.1. При размещении элементов во вспомогательных формах рекомендуется действовать следующим образом:

1. Среди всего множества реквизитов и элементов выбрать второстепенные, в большинстве случаев заполняемые автоматически, а также те, которые использует небольшое количество пользователей. Чаще всего такие реквизиты будут являться необязательными для заполнения
2. Разделить эти элементы на несколько смысловых групп
3. Разместить эти группы в одной или нескольких вспомогательных формах

3.2. Для открытия вспомогательной формы следует использовать гиперссылку, размещаемую в основной форме. При открытии вспомогательная форма блокирует окно владельца.

3.3. Текст гиперссылки может формироваться одним из двух способов.

Вариант 1. В тексте ссылки указывается название вспомогательной формы. При этом название должно быть таким, чтобы по нему быть понятно, какие реквизиты могут располагаться внутри.

Например:

The screenshot shows a Windows application window titled "Поступление товаров и услуг (создание) (Товары)". At the top, there are buttons for "Провести и закрыть", "Запись", "Провести", "Печать", and "Еще". Below these are fields for "Накладная №", "Организация" (set to "Ромашка ООО"), "Номер" (auto-filled with "19.11.2013 0:00:00"), "Склад", "Контрагент", and "Подразделение". A modal dialog box titled "Грузоотправитель и грузополучатель" is open. It contains two sections: "Грузоотправитель" (with "Контрагент" selected) and "Грузополучатель" (with "Ромашка ООО" selected). At the bottom of the dialog are "OK" and "Отмена" buttons. In the main window, below the dialog, there are fields for "Счет-фактура" (set to "Не требуется"), "Всего" (0.00 руб.), "НДС (в т.ч.)" (0.00), "Комментарий" (empty), and "Ответственный" (set to "Ромашкина Тамара Петровна"). There are also links for "Как настроить счета расчетов с контрагентами" and "Как настроить учет номенклатуры".

Вариант 2. В тексте ссылки указывается перечень значений реквизитов, заполняемых внутри вспомогательной формы. Если этот перечень слишком велик, то в первую очередь в него следует вынести реквизиты:

- значения которых важно видеть сразу из основной формы
- которые обязательны для заполнения

Например:

Если текст ссылки выводится в виде перечня значений, то для нее следует добавлять заголовок. В табличной части заголовком будет являться название колонки.

Правильно

Расчеты: [62.01, 62.02, зачет аванса автоматически](#)
Цены и валюта: [Цены с НДС. Основная цена покупки](#)

Неправильно

Расчеты: [62.01, 62.02, зачет аванса автоматически](#)
Цены и валюта: [Цены с НДС. Основная цена покупки](#)

3.4. Если внутри вспомогательной формы есть реквизиты, обязательные для заполнения, и они не заполнены, то для гиперссылки используется красный цвет (элемент стиля вида цвет "НезаполненныйРеквизит", RGB 178,34,34) Например:

3.5. Командная панель во вспомогательной форме состоит из двух кнопок: "OK" и "Отмена" и располагается в нижней части формы:

4. Свертываемые группы

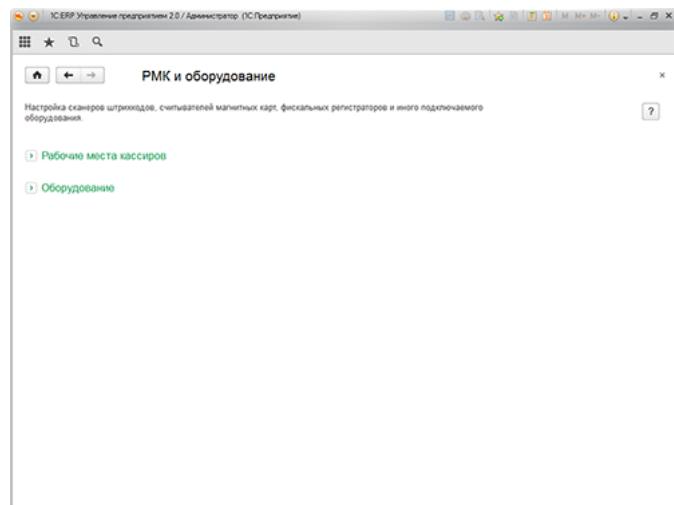
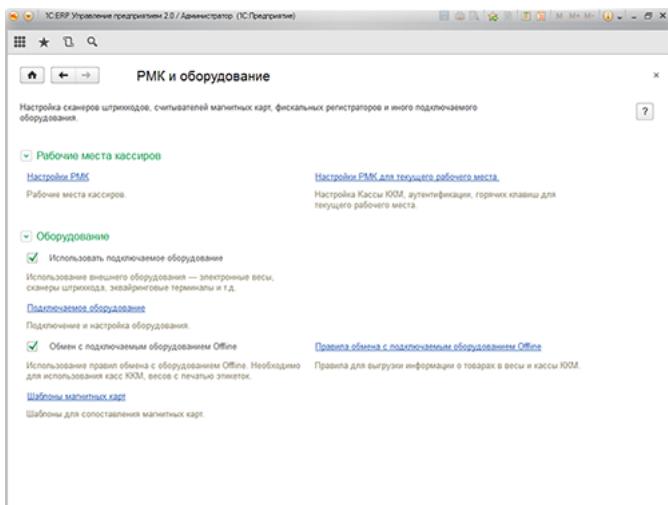
4.1. Свертываемые группы являются разделами реквизитов или настроек, логически связанных между собой и объединенных общим заголовком. Возможность свертывания групп позволяет упростить поиск нужного раздела и информации, которая в нем расположена.

4.2. Формы, вся информация в которых размещена в свертываемых группах, рекомендуется отображать с учетом быстрого доступа к содержимому разделов:

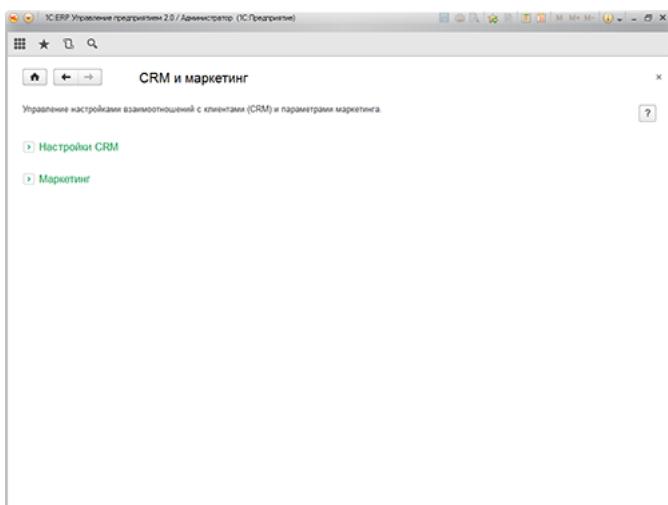
- если все группы в развернутом виде отображаются на странице без вертикальной прокрутки – раскрывать эти группы по умолчанию;
- если первая группа развернута, а все последующие группы в свернутом виде отображаются на странице без вертикальной прокрутки – раскрывать первую группу по умолчанию.

Правильно

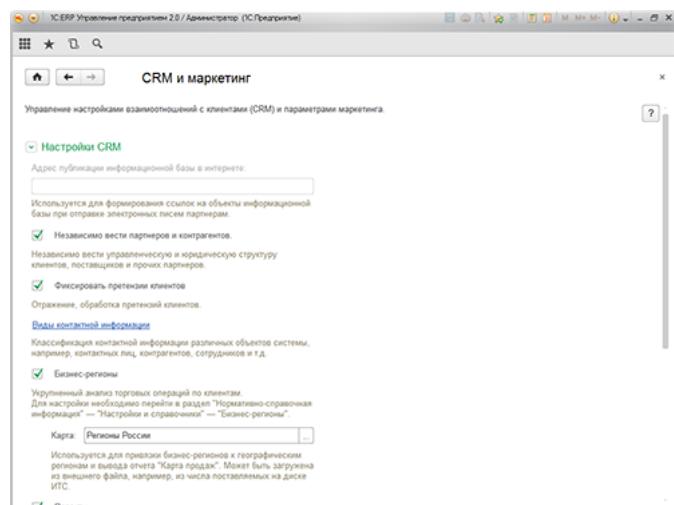
Неправильно



Правильно



Неправильно



См. также:

[Компоновка форм \(8.2\)](#)

[Оформление элементов \(8.2\)](#)

[Переход к форме с дополнительными реквизитами \(8.2\)](#)

Командная панель документа

Область применения: управляемое приложение.

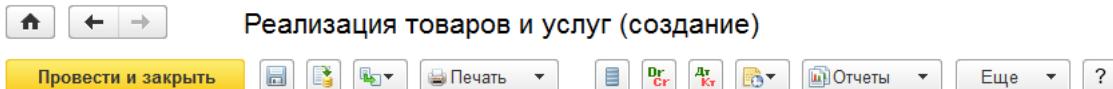
#std716

1. В командной панели кнопка по умолчанию должна быть расположена самой крайней слева. В подавляющем большинстве случаев кнопкой по умолчанию является "Провести и закрыть" или "Записать и закрыть"
2. Порядок расположения команд во всех документах должен быть одинаковым.
3. Состав системных кнопок командной панели, отображаемых платформой по умолчанию, и их порядок относительно друг друга изменять не рекомендуется.
4. Командная панель должна позволять пользователю при стандартных настройках экрана (ширина экрана 1280 точек, панель инструментов выведена слева вертикально) выполнить самые важные и частотные действия с документом, не открывая подменю "Еще".
5. При стандартных настройках экрана пользователю должны быть сразу видны все важные команды:
 - Провести и закрыть / Записать и закрыть
 - Записать
 - Провести
 - Создать на основании
 - Печать
 - Глобальные команды
 - Дополнительные сведения
 - Движения документа
 - и т.д.

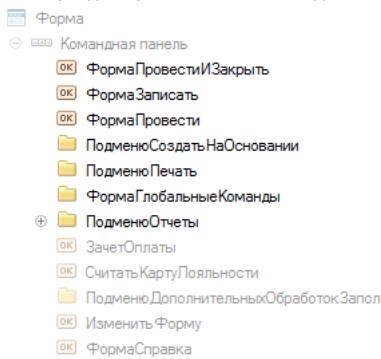
- Отчеты (контекстные, из панели навигации)

6. Если в командной панели расположено большое количество команд, рекомендуется отображать важные команды кнопками с картинками без текста.

Например, командная панель документа "Реализация товаров и услуг":



В Конфигураторе элементы командной панели структурируются следующим образом:



Команда "Записать" — свойство Отображение = Картинка

Команда "Провести" — свойство Отображение = Картинка

Подменю "Создать на основании" — свойство Отображение = Картинка

Подменю "Печать" — свойство Отображение = Авто

Подменю "Отчеты" — свойство Отображе

Командная панель формы (8.2)

Табличные части. Оформление списка

Область применения: управляемое приложение.

#std717

1. Шапка табличной части должна состоять из одной строки и для "одноэтажных" списков, и для "многоэтажных".

В "многоэтажных" колонках следует предусмотреть общий заголовок, соответствующий смыслу указываемых в них данных, а также отображение подсказки в ячейках с незаполненными значениями.

Для отображения подсказки следует одновременно использовать:

- элемент стиля вида шрифт "ПодсказкаНезаполненнойЧекой" (размер: 8, начертание: наклонный)
- свойство "ПодсказкаВвода"

Например, для реквизитов с кратким и полным названиями номенклатуры выводится общий заголовок колонки – "Номенклатура", а в ячейках выводятся подсказки:

N	Номенклатура	Количество	Цена
1	<p>Наименование</p> <p>Содержание услуги</p>		

В Конфигураторе отображение заголовков колонок в шапке отключается с помощью свойства "Отображать в Шапке".

2. Заголовок колонки должен полностью помещаться в шапку

3. Если в табличной части много колонок и они заведомо не помещаются при стандартном разрешении экрана без горизонтальной прокрутки, то часть из них следует расположить первыми и зафиксировать.

Фиксировать нужно самые важные колонки, которые позволят идентифицировать объект. Колонка с номером по порядку также фиксируется.

Например, для списка сотрудников фиксируется колонка "ФИО", для списка товаров и услуг – колонка "Номенклатура", для списка основных средств – наименование основного средства.

Все остальные реквизиты будут находиться в области, скроллируемой горизонтально.

Все остальные
Например:

N	Номенклатура	НДС	Всего	Счета учета
1	Юридические услуги	15,25	100,00	90.01.1. Основная номенклатурная группа_90.02.1. 90.03
	Юридические услуги по регистрации			

В Конфигураторе фиксация колонки табличной части осуществляется с помощью свойства "Фиксация в Таблице" (=лево)

4. Колонкам, предназначенным для указания цифровых значений, рекомендуется явно указывать ширину, достаточную для указания наиболее частотных значений. При необходимости большей ширины пользователь сможет расширить колонку самостоятельно.

Например, для бухгалтерских документов в большинстве ситуаций достаточно ширины 10 (эта ширина позволяет указывать суммы в несколько млн руб.).

Итоги в документах

Область применения: управляемое приложение.

#std718

Итоги в документах можно выводить несколькими способами:

- отдельными полями ввода
- в подвалах таблиц

1. Итоги в виде отдельных полей ввода

Расположение

1.1. Итоги размещаются сразу под таблицей (или несколькими таблицами), по которым они выводятся. Не следует вставлять элементы, разделяющие табличную часть и группу с итогами.

Правильно

Счет на оплату покупателю 2 от 12.11.2013

Номер	Контрагент	Организация	Банковский счет	Договор	Подразделение
2	ЗАО "Айтис"	Ромашка ООО			

Цены с НДС

Товары (2)	Услуги	Дополнительно					
Добавить	Подбор	Изменить					
Еще							
N	Номенклатура	Количество	Цена	Сумма	% НДС	НДС	Всего
1	Комплект постельного белья	10.000	500.00	5 000.00	18%	762.71	5 000.00
2	Комплект постельного белья	5.000	600.00	3 000.00	18%	457.63	3 000.00

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34

Комментарий: Ответственный: Ромашкина Тамара Петровна

Как вывести ответственных лиц организаций в документах

Неправильно

Счет на оплату покупателю 2 от 12.11.2013

Номер	Контрагент	Организация	Банковский счет	Договор	Подразделение
2	ЗАО "Айтис"	Ромашка ООО			

Цены с НДС

Товары (2)	Услуги	Дополнительно					
Добавить	Подбор	Изменить					
Еще							
N	Номенклатура	Количество	Цена	Сумма	% НДС	НДС	Всего
1	Комплект постельного белья	10.000	500.00	5 000.00	18%	762.71	5 000.00
2	Комплект постельного белья	5.000	600.00	3 000.00	18%	457.63	3 000.00

Комментарий: Ответственный: Ромашкина Тамара Петровна

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34

Как вывести ответственных лиц организаций в документах

1.2. В области итогов следует размещать только те элементы, которые относятся к итоговым данным. Иные элементы следует располагать вне области итогов.

Например, в форме "Акта сверки расчетов" реквизит "Остаток на начало" не является итоговым значением, а приводится информационно. Поэтому его не следует включать в группу с итогами и выделять серым фоном.

Правильно

Остаток на начало:	0,00	Остаток на конец:	0,00	Расхождение с контрагентом:	0,00
--------------------	------	-------------------	------	-----------------------------	------

Неправильно

Остаток на начало:	0,00	Остаток на конец:	0,00	Расхождение с контрагентом:	0,00
--------------------	------	-------------------	------	-----------------------------	------

1.3. Область с итогами выравнивается по правому краю формы

Правильно

Неправильно

N	Номенклатура	Количество	Цена	Сумма	% НДС	НДС	Всего
1	Комплект постельного белья	10,000	500,00	5 000,00	18%	762,71	5 000,00
2	Комплект постельного белья	5,000	600,00	3 000,00	18%	457,63	3 000,00

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34

Комментарий: [] Ответственный: Ромашкина Тамара Петровна
Как вывести ответственных лиц организации в документах
Как создать счет на оплату

N	Номенклатура	Количество	Цена	Сумма	% НДС	НДС	Всего
1	Комплект постельного белья	10,000	500,00	5 000,00	18%	762,71	5 000,00
2	Комплект постельного белья	5,000	600,00	3 000,00	18%	457,63	3 000,00

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34

Комментарий: [] Ответственный: Ромашкина Тамара Петровна
Как вывести ответственных лиц организации в документах
Как создать счет на оплату

Оформление

1.4. Итоги выделяются серым фоном. Для этого используется элемент стиля "ИтогиФонГруппы" (RGB 220,220,220)

1.5. Итоги оформляются отдельными полями ввода с установленным признаком «ТолькоПросмотр»

1.6. В заголовках полей текст выводится без слова «итог»

1.7. Валюта (в случае отражения) выводится после поля ввода. Если итоги включают в себя несколько взаимосвязанных полей, то валюта выводится только после первого из них.

Например, в области итогов, содержащих взаимосвязанные поля "Всего" и "НДС (в т.ч.)" валюта выводится только после поля "Всего".

Правильно

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34

Неправильно

Всего: 8 000,00 руб. НДС (в т.ч.): 1 220,34 руб.

2. Итоги в подвалах таблиц

2.1. В подвалах таблиц итоги следует располагать в случае, если итоговые значения выводятся по значительному числу колонок (более 4) одной табличной части и при оформлении в виде отдельных полей ввода не помещаются на форму в одну строку.

Итоги в документах (8.2)

Поля "Ответственный" и "Комментарий"

Область применения: управляемое приложение.

#std719

1. Поле "Комментарий"

Поле "Комментарий" может выводиться:

- В виде одностороннего поля в нижней части формы документа
- В виде многострочного поля на отдельной вкладке формы документа

Если поле "Комментарий" в типовых случаях будет большим, то допускается делать его многострочным и выносить на отдельную вкладку. В остальных случаях рекомендуется использовать одностороннее поле "Комментарий".

1.1. Одностороннее поле "Комментарий"

- Располагается в нижней части формы документа, самым последним
- Заголовок располагается слева от поля
- В поле отсутствуют кнопки выбора, списка выбора и другие

Комментарий: []

1.2. Многострочное поле "Комментарий"

- Располагается на отдельной вкладке
- Заголовок "Комментарий" выводится в названии вкладки. У поля заголовок не отображается
- На вкладке размещается только поле с текстом комментария, которое растягивается по вертикали и по горизонтали
- Если поле с комментарием заполнено, то в заголовке вкладки отражается картинка "Комментарий"

Комментарий

Выставлен счет на оплату

2. Поле "Ответственный"

- Поле следует выводить в нижней части формы документа.

3. Группа полей "Ответственный" и "Комментарий"

- При одновременном наличии в форме документа поля "Ответственный" и однострочного поля "Комментарий", их следует объединять в группу.
- Эту группу следует располагать в самом низу формы документа.
- Внутри группы поля размещаются в одну строку, рядом друг с другом.

The screenshot shows a window titled 'Приходный кассовый ордер (создание)' (Cash Receipt Order (Creation)). At the top, there are buttons for 'Home', 'Back', 'Forward', 'Close', 'Print', 'Save', 'Create based on', and 'More'. Below the buttons, there are two input fields: 'Комментарий:' (Comment) and 'Ответственный:' (Responsible Person). The 'Responsible Person' field contains 'Любимов Валерий Юрьевич' (Valeriy Yuryevich Lyubimov). At the bottom of the window, there is another set of input fields: 'Комментарий:' and 'Ответственный:', both currently empty. This illustrates how the two fields are grouped together at the bottom of the form.

См. также:

[Реквизит "Комментарий" у документов](#)

[Поля "Ответственный" и "Комментарий" \(8.2\)](#)

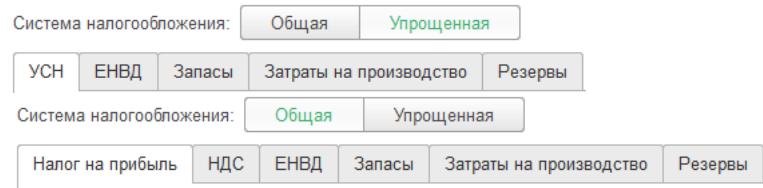
Тумблер

Область применения: управляемое приложение.

#std720

1. Тумблер следует использовать в ситуациях, когда при выборе значения происходит изменение состава или расположения элементов в форме. Он похож на кнопку и своим видом показывает, что при нажатии что-то произойдет.

Например, при выборе системы налогообложения меняется состав вкладок:



2. Количество кнопок в тумблере рекомендуется делать небольшим, а их названия - краткими. Если количество вариантов - большое, а названия длинные, то лучше использовать выпадающий список.

3. Заголовок рекомендуется добавлять в случае, если:

- по названиям кнопок не понятно назначение тумблера
- тумблер выступает в качестве переключателя

Подсказки на форме

Область применения: управляемое приложение.

#std721

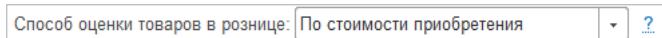
Подсказки рекомендуется выводить по тем элементам интерфейса, которые требуют пояснения, расшифровки и донесения до пользователя подробного описания их назначения. Для реквизитов, используемых в ежедневной работе, подсказки добавлять не следует.

В тексте подсказки не рекомендуется приводить инструкции и объяснять работу реквизита, вместо этого следует оптимизировать сценарии работы, с ним связанные.

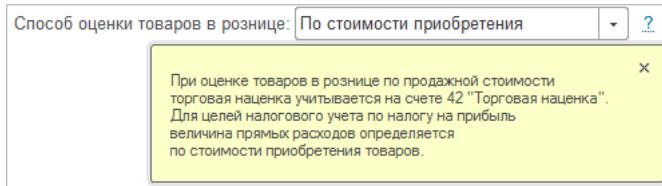
1. Текстовые подсказки к полям

1.1. Текстовые подсказки к полям рекомендуется прятать в гиперссылке "?" Для этого в Конфигураторе устанавливается режим отображения подсказки

"Кнопка".

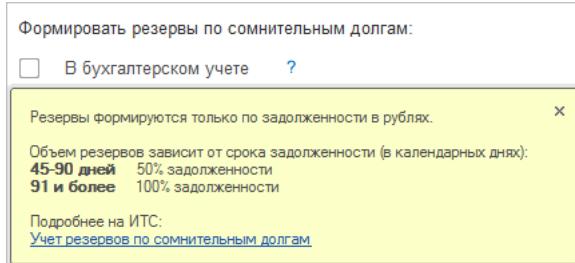


При нажатии на ссылку "?" отображается текст подсказки:



1.2. Если в подсказку, помимо обычного текста, необходимо добавить ссылки, иконки или использовать форматированный текст, то следует использовать "расширенную" подсказку.

Пример расширенной подсказки:

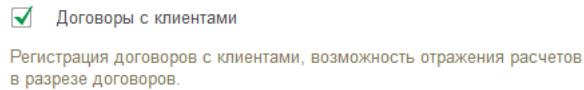


1.3. Не рекомендуется выводить в подсказке много текста (более 255 символов, включая пробелы). Если потребуется более длинная подсказка, то в ней можно дать ссылку на статью ИТС с подробным описанием.

2. Текстовые подсказки к настройкам

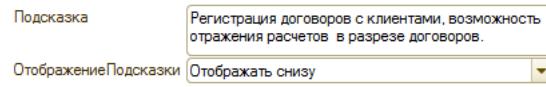
2.1. Текстовые подсказки к настройкам следует выводить внизу — под настройкой.

Пример подсказки:



2.2. В конфигураторе подсказки следует оформлять с помощью настройки "Подсказка" с параметром отображения "Отображать снизу".

Пример оформления подсказки:



Допускается так же использование расширенных подсказок. Использование в качестве текстовых подсказок декорации "Надпись" не рекомендуется.

Пример:

<p>Правильно</p> <ul style="list-style-type: none">⊕ ГруппаИспользоватьДоговорыСКлиентами<ul style="list-style-type: none">⊖ ИспользоватьДоговорыСКлиентами<ul style="list-style-type: none">⊕ Расширенная подсказка	<p>Неправильно</p> <ul style="list-style-type: none">⊕ ГруппаИспользоватьДоговорыСКлиентами<ul style="list-style-type: none">⊖ ИспользоватьДоговорыСКлиентами<ul style="list-style-type: none"><input checked="" type="checkbox"/> ПояснениеИспользоватьДоговорыСКлиентами
--	--

Использование декораций "Надпись" для подсказок допустимо в случаях, когда нужное поведение нельзя реализовать иначе. Например, подсказки к полям ввода, которые должны быть выровнены вместе со всем элементом, включая заголовок.

См. также

- [Подсказка и проверка заполнения](#)

Шрифты

Область применения: управляемое приложение.

#std687

Действует для платформы с версии 8.3.2

При разработке конфигураций следует использовать стилевые шрифты:

<Шрифт текста>
<Мелкий шрифт текста>
<Крупный шрифт текста>
<Очень крупный шрифт текста>

При необходимости также можно использовать шрифты из состава MS Core Fonts:

Andale Mono
Arial
Arial Black

Comic Sans MS
Courier New
Georgia
Impact
Times New Roman
Trebuchet MS
Verdana
Webdings

Не следует использовать в конфигурациях прочие шрифты, так как они могут поддерживаться не во всех операционных системах, в которых работает клиент 1С:Предприятия.

Пользовательские представления объектов

Область применения: управляемое приложение.

#std468

Если для объекта метаданных определены свойства пользовательского представления, они заполняются следующим образом.

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

См. также: [Имя, синоним, комментарий, Тексты](#)

2. **Представление объекта (для регистра - представление записи)**. Задается название объекта в единственном числе, например, "Валюта". Название должно быть лаконичным и понятным. Например, вместо "Версия проверяемой конфигурации" нужно использовать "Версия".

Представление объекта заполняется в случае, если синоним не может быть использован, как название объекта в единственном числе.

3. **Расширенное представление объекта (для регистра - расширенное представление записи)**. Задается полное название объекта в единственном числе. Заполняется в случае, когда название объекта, заданное в представлении объекта (или, если не заполнено, то - в синониме) менее информативно, чем его полное название. Например, в расширенном представлении объекта указано "Реализация товаров и услуг", в то время как в представлении объекта указано "Реализация".

4. **Представление списка**. Задается название объектов во множественном числе, например, "Валюты". Кроме этого, в некоторых случаях может указываться название списка, если оно является самостоятельным термином, например, «Классификатор единиц измерения». Название должно быть лаконичным и понятным. Например, вместо "Общероссийский классификатор основных фондов" нужно использовать "Классификатор ОКОФ".

Заполняется в случае, если синоним не может быть использован, как название списка объектов.

5. **Расширенное представление списка**. Задается полное название списка объектов. Заполняется в случае, когда заданное в представлении списка (или, если не заполнено, то в синониме) название списка менее информативно, чем полное название списка.

Например, в расширенном представлении списка указано "Номенклатура (товары и услуги)", представление списка не заполнено, а в синониме указано "Номенклатура".

6. **Расширенное представление**. Задается полное название объекта метаданных. Заполняется в случае, когда заданное в синониме название объекта менее информативно, чем его полное название.

Например, синоним обработки - "Клиент банка", а расширенное представление - "Клиент банка (обмен платежными документами)".

7. **Пояснение**. Задается пояснение по назначению данного объекта метаданных. Пояснение задается в виде законченных предложений. Заполняется для объектов метаданных, представление которых не достаточно точно передает их назначение.

Например, для справочника "Организации" пояснение может быть задано так: "Юридические лица и предприниматели нашей компании.".

8. **Картинка**. Задается для подсистем верхнего уровня с установленным флагком "Включать в командный интерфейс". Размер картинки: 32 на 32.

См. также

- [Приложение 3. Правила формирования текстов стандартных команд и автоматических заголовков форм](#) в документации к платформе 1С:Предприятие.

Использование сочетаний клавиш, список зарезервированных сочетаний

Область применения: управляемое приложение.

#std430

Для часто выполняемых или общеупотребимых действий рекомендуется назначать сочетания клавиш. Ниже приведены списки зарезервированных сочетаний в прикладных решениях и в платформе 1С:Предприятие.

Для действий, приведенных в таблицах, назначение указанных сочетаний клавиш обязательно. Использование зарезервированных сочетаний в иных целях запрещается.

Список зарезервированных сочетаний клавиш в прикладных решениях:

Сочетание клавиш Действие

Ctrl + Shift + F	Полнотекстовый поиск
F7	Поиск по штрихкоду
F8	Выключить/включить активность проводок в журнале операций
F11	Установить текущее учреждение
F12	Вызов "Управление данными работника"

Список зарезервированных сочетаний клавиш в платформе 1С:Предприятие:

Сочетание клавиш Действие

Ctrl + A	Выделить все
Ctrl + B	Включить/выключить жирность
Ctrl + Alt + B	Предыдущая страница
Ctrl + C	Копировать в буфер обмена
Ctrl + E	Открыть "Свойства"
Ctrl + F	Поиск, найти
Ctrl + Alt + F	Следующая страница
Shift + Alt + F	Форматировать блок
Ctrl + G	Перейти к ячейке, перейти к строке
Ctrl + H	Заменить, замена
Ctrl + I	Включить/выключить курсив

Ctrl + L	Удалить текущую строку
Ctrl + Shift + M	Перенести элемент в другую группу
Ctrl + N	Создать новый документ
Ctrl + Shift + N	Установка имени текущей области
Ctrl + O	Открыть существующий документ
Ctrl + Alt + O	Открыть "Служебные сообщения"
Ctrl + P	Печать активного документа
Ctrl + Shift + P	Печать на текущий принтер
Ctrl + Shift + R	Обновить, обновить группировки
Shift + Alt + R	Восстановить положение окна
Alt + Shift + R	Восстановить положение окна
Ctrl + S	Сохранить активный документ
Ctrl + T	Найти в дереве
Ctrl + U	Включить/выключить подчеркивание
Ctrl + V	Вставить из буфера обмена
Ctrl + W	Выделить слово
Ctrl + Alt + W	Открыть "Табло"
Ctrl + X	Вырезать в буфер обмена
Ctrl + Y	Вернуть отмененное действие
Ctrl + Z	Отменить последнее действие
Ctrl + Alt + Z	Очистить служебные сообщения
Ctrl + Shift + Z	Закрыть "Служебные сообщения"
Ctrl + "-"	Переход по истории активности окон, переместиться назад
Ctrl + Shift + "-"	Переход по истории активности окон, переместиться вперед
Ctrl + Alt + Num-	Свернуть (узел дерева, группу табличного документа) и все подчиненные
Shift + Num-	Вычертить из буфера обмена
Alt + Num-	
Space	Изменение флашка
Alt + Space	Вызвать системное меню приложения или модального диалога
Alt + Hyphen (-)	Вызвать системное меню окна (кроме модальных диалогов)
Enter	Перейти к редактированию содержимого ячейки, сохранить свойства
Alt + Enter	Открыть "Свойства"
Ctrl + Enter	Сформировать отчет
F1	Открыть "Справку"
Shift + F1	Открыть "Содержание справки"
Shift + Alt + F1	Открыть "Индекс справки"
Alt + F1	Открыть "Поиск по справке"
F2	Открыть, переключение режима редактирования/ввода в ячейке
Ctrl + F2	Открыть встроенный "Калькулятор" системы 1С:Предприятие
Shift + F2	Закончить редактирование, предыдущая закладка
Alt + F2	Установить/снять закладку
F3	Найти следующий
Ctrl + F3	Найти следующий выделенный
Shift + F3	Найти предыдущий
Ctrl + Shift + F3	Найти предыдущий выделенный
F4	Кнопка выбора
Ctrl + F4	Закрыть активное обычное окно
Shift + F4	Очистить поле
Alt + F4	Закрыть активное окно, модальный диалог или приложение
Ctrl + Shift + F4	Кнопка открытия
F6	Активизировать следующую секцию окна
Ctrl + F6	Активизировать следующее обычное окно
Shift + F6	Активизировать предыдущую секцию окна
Ctrl + Shift + F6	Активизировать предыдущее обычное окно
F9	Скопировать
Ctrl + F9	Новая группа
F10	Вызвать главное меню
Shift + F10	Вызвать контекстное меню
Alt + F10	Вызвать главное меню
Up, Down, Left, Right	Перемещение по ячейкам, прокрутить, перейти к предыдущему / следующему свойству
Ctrl + (Up, Down, Left, Right)	Перемещение по ячейкам или дереву, перемещение выделенного варианта, перемещение по тексту по словам
Shift + (Up, Down, Left, Right)	Выделение ячеек, изменение размеров элемента карты
Alt + Right	Переход к следующей главе справки
Alt + Left	Переход к предыдущей главе справки
Ctrl + Shift + (Up, Down, Left, Right)	Перемещение по ячейкам к следующей заполненной или пустой с выделением ячеек, выделение колонок, выделение слов, перемещение строки верх/вниз
Alt + Shift + (Up, Down)	Выделение строк
Ctrl + Alt + Shift + (Left, Right)	Выделение колонок до следующей заполненной или пустой ячейки
Ctrl + Alt + Shift + (Up, Down)	Выделение строк до следующей заполненной или пустой ячейки
Page Up	Прокрутить на страницу вверх
Ctrl + Page Up	Предыдущая страница, перейти к предыдущей категории
Shift + Page Up	Выделить предыдущую страницу текста
Alt + Page Up	Прокрутить на размер окна влево, прокрутить на страницу влево
Page Down	Прокрутить на страницу вниз
Ctrl + Page Down	Следующая страница, перейти к следующей категории
Shift + Page Down	Выделить следующую страницу текста
Alt + Page Down	Прокрутить на размер окна вправо, прокрутить на страницу вправо
Num+	Раскрыть узел дерева, увеличить масштаб, раскрыть категорию свойств
Ctrl + Num+	Развернуть (узел дерева, группу табличного документа, группировку модуля)
Shift + Num+	Добавить к буферу обмена
Ctrl + Shift + Num+	Развернуть (все узлы дерева, группы табличного документа, группировки модуля)
Ctrl + Alt + Num+	Развернуть (узел дерева, группу табличного документа, группировку модуля) и все подчиненные
Num-	Закрыть узел дерева, уменьшить масштаб, закрыть категорию свойств

Ctrl + Num-	Свернуть (узел дерева, группу табличного документа, группировку модуля)
Shift + Num-	Вычертить из буфера обмена
Alt + Num-	Вызвать системное меню окна (кроме модальных диалогов)
Ctrl + Shift + Num-	Свернуть (все узлы дерева, группы табличного документа, группировки модуля)
Ctrl + Alt + Num-	Свернуть (узел дерева, группу табличного документа, группировку модуля) и все подчиненные
Num*	Раскрыть все узлы дерева
Shift + Num*	Копировать в буфер обмена как число
Esc	Вернуть активность обычному окну, снять выделение, восстановить значения свойства
Shift + Esc	Закрыть активное окно (кроме обычных)
Tab	Сдвинуть блок вправо, перейти на следующий элемент карты
Ctrl + Tab	Активизировать следующее обычное окно
Shift + Tab	Перейти на предыдущий элемент карты, сдвинуть блок влево
Ctrl + Shift + Tab	Активизировать предыдущее обычное окно
BackSpace	Удалить символ слева от курсора
Ctrl + BackSpace	Удалить слово слева от курсора
Alt + BackSpace	Отменить последнее действие
Shift + Alt + BackSpace	Вернуть отмененное действие
Ins	Добавить, переключить режим вставки/замены
Ctrl + Ins	Копировать в буфер обмена
Shift + Ins	Вставить из буфера обмена
Del	Удалить
Ctrl + Del	Удалить слово справа от курсора
Shift + Del	Вырезать в буфер обмена, удалить строку
Home	Перейти в начало строки, перейти в начало палитры
Ctrl + Home	Перейти в начало текста
Shift + Home	Выделить до начала строки
Ctrl + Shift + Home	Выделить до начала текста
End	Перейти в конец строки
Ctrl + End	Перейти в конец текста
Shift + End	Выделить до конца строки
Ctrl + Shift + End	Выделить до конца текста
Alt	Вызвать главное меню
Alt + Hyphen (-)	Вызвать системное меню окна (кроме модальных диалогов)
Ctrl + [Перейти по операторным скобкам назад
Ctrl + Shift + [Перейти по операторным скобкам назад с выделением текста
Ctrl +]	Перейти по операторным скобкам вперед
Ctrl + Shift +]	Перейти по операторным скобкам вперед с выделением текста
Ctrl + Break	Вызвать главное меню
Alt	Вызвать главное меню

Длительные операции на сервере

Область применения: управляемое приложение.

#std642

1. При разработке конфигураций следует избегать длительных вызовов из клиентского кода в серверный. Все длительные серверные вызовы, которые могут выполняться более 8 секунд в обычных сценариях работы пользователя, следует выполнять асинхронно, с помощью фонового задания.

К таким операциям относятся: формирование отчета, групповая обработка объектов, загрузка или выгрузка данных в другое приложение, заполнение больших табличных частей и т.п.

В противном случае такие вызовы могут привести к потере работоспособности приложения или затруднению работы с ним:

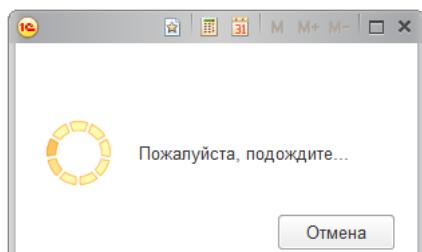
- браузер может предложить прекратить длительно выполняющийся сценарий, после чего приложение станет неработоспособным;
- веб сервер может прервать длительное обращение к серверу 1С:Предприятия и вернуть ошибку 504 (шлюз не отвечает);
- в случае длительного выполнения операции, у пользователя нет возможности отменить ее.

2.1. Общий подход к асинхронному выполнению длительных серверных операций с помощью фонового задания:

- Код, выполняющий длительную обработку данных, располагается в модуле менеджера объекта* или в общем модуле. Результат своей работы он помещает во временное хранилище;

*Примечание: необходимо использовать процедуру-обертку в общем модуле, которая будет вызывать процедуру модуля менеджера через **ОбщегоНазначения.ВыполнитьМетодКонфигурации()**. Т.к. фоновые задания могут работать только с процедурами и функциями общих модулей.

- Для выполнения этого кода на сервере запускается фоновое задание, при этом необходимо ожидать завершения выполнения фонового задания в течение **0.8** сек;
- Если за время ожидания выполнения задания оно не завершилось, то управление возвращается на клиент, и в клиентском коде подключается обработчик ожидания, в котором периодически проверяется состояние фонового задания. При этом интервал опроса задания увеличивается от **1** до **15** секунд с фиксированным коэффициентом **1.4**;
- На время выполнения длительной операции пользователю отображается индикатор; при этом для отчетов индикатор выводится в поле табличного документа, используя свойство поля табличного документа **ОтображениеСостояния**:



- При получении от сервера информации о том, что фоновое задание завершено, полученный результат загружается из временного хранилища и обрабатывается.

2.2. Асинхронное формирование отчета требуется только для тех отчетов, которые

- разработаны без использования СКД или с использованием СКД, но с переопределенной процедурой формирования отчета (переопределен обработчик кнопки «Сформировать» или в обработчике модуля отчета **ПриКомпоновкеРезультата** устанавливается **СтандартнаяОбработка = Ложь**).
- и формирование которых, как правило, занимает длительное время.

Поведение таких отчетов должно быть максимально похожим на поведение отчетов на базе СКД, а именно:

- форму отчета не следует блокировать на время его формирования;
- пользователь может изменить настройки и переформировать отчет, не дожидаясь окончания его формирования;
- при закрытии формы отчета, формирование отчета прерывается.

3. При использовании в конфигурации **Библиотеки стандартных подсистем** в распоряжении разработчика имеются вспомогательные функции и процедуры общих модулей **ДлительныеОперации**, **ДлительныеОперацииКлиент**, а также процедура **УстановитьСостояниеПоляТабличногоДокумента** общего модуля **ОбщегоНазначенияКлиентСервер**.

Пример выполнения функции в фоновом задании при использовании в конфигурации Библиотеки стандартных подсистем. В модуле менеджера объекта размещена функция, которая выполняет поиск настроек и возвращает их:

Функция ОпределитьНастройкиУчетнойЗаписи(АдресЭлектроннойПочты, Пароль) Экспорт

...
Возврат Настройки;
КонецФункции

В форме объекта выполняется вызов этой функции в фоновом задании в три этапа:

- 1) запуск фонового задания на сервере,
- 2) подключение обработчика завершения фонового задания на клиенте,
- 3) обработка результата выполнения фонового задания.

&НаКлиенте

Процедура НастроитьПараметрыПодключенияАвтоматически()

// 1. Запуск фонового задания на сервере.
ДлительнаяОперация = НачатьПоискНастроекУчетнойЗаписи();

// 2. Подключение обработчика завершения фонового задания.

ПараметрыОжидания = ДлительныеОперацииКлиент.ПараметрыОжидания(ЭтотОбъект);
Оповещение = Новый ОписаниеОповещения("ПриЗавершенииПоискаНастроек", ЭтотОбъект);
ДлительныеОперацииКлиент.ОжидатьЗавершение(ДлительнаяОперация, Оповещение, ПараметрыОжидания);
КонецПроцедуры

&НаСервере

Функция НачатьПоискНастроекУчетнойЗаписи()

ПараметрыВыполнения = ДлительныеОперации.ПараметрыВыполненияФункции(УникальныйИдентификатор);
Возврат ДлительныеОперации.ВыполнитьФункцию(ПараметрыВыполнения,
"Справочники.УчетныеЗаписиЭлектроннойПочты.ОпределитьНастройкиУчетнойЗаписи",
АдресЭлектроннойПочты, Пароль);
КонецФункции

// 3. Обработка результата выполнения фонового задания.

&НаКлиенте

Процедура ПриЗавершенииПоискаНастроек(Результат, ДополнительныеПараметры) Экспорт

Если Результат = Неопределено Тогда // Пользователь отменил задание.
Возврат;
КонецЕсли;

Если Результат.Статус = "Ошибка" Тогда
ВызватьИсключение Результат.КраткоеПредставлениеОшибка;
КонецЕсли;

Настройки = ПолучитьИзВременногоХранилища(Результат.АдресРезультата);
удалитьИзВременногоХранилища(Результат.АдресРезультата);
УстановитьНастройкиУчетнойЗаписи(Настройки);

КонецПроцедуры

Методическая рекомендация (полезный совет)

3.1. При каждом выполнении фонового задания его результат помещается во временное хранилище на время жизни формы:

ПараметрыВыполнения = ДлительныеОперации.ПараметрыВыполненияФункции(УникальныйИдентификатор);
ДлительныеОперации.ВыполнитьФункцию(ПараметрыВыполнения, ПараметрФоновогоЗадания);

Если длительная операция выполняется пользователем многократно, пока эта форма открыта, то временные хранилища накапливаются, что вызывает рост потребления памяти. Поэтому для уменьшения расхода оперативной памяти в большинстве случаев рекомендуется очищать временное хранилище сразу после получения результата фонового задания:

Настройки = ПолучитьИзВременногоХранилища(Результат.АдресРезультата);
УдалитьИзВременногоХранилища(Результат.АдресРезультата); // данные во временном хранилище больше не требуются.

Если же результат фонового задания требуется сохранять на протяжении нескольких серверных вызовов, то необходимо передавать фиксированный адрес заранее инициализированного временного хранилища:

&НаСервере

Процедура ПриСозданииНаСервере(Отказ)

АдресРезультатаФоновогоЗадания = ПоместитьВоВременноеХранилище(Неопределено, УникальныйИдентификатор); // Резервируем адрес временного хранилища
КонецПроцедуры

&НаСервере

Функция НачатьПоискНастроекУчетнойЗаписи()

ПараметрыВыполнения = ДлительныеОперации.ПараметрыВыполненияФункции(УникальныйИдентификатор);
ПараметрыВыполнения.АдресРезультата = АдресРезультатаФоновогоЗадания; // всегда используем одно и то же временное хранилище

Возврат ДлительныеОперации.ВыполнитьФункцию(ПараметрыВыполнения,
"Справочники.УчетныеЗаписиЭлектроннойПочты.ОпределитьНастройкиУчетнойЗаписи",
АдресЭлектроннойПочты, Пароль);
КонецФункции

4. Если в конфигурации реализуются алгоритмы, инициирующие запуск фоновых заданий или запись данных информационной базы **без участия пользователя** (например, регулярное обновление информации в открытой форме), то в них следует проверять, что в текущем сеансе не установлен монопольный режим. В противном случае, следует блокировать попытки выполнения таких действий. Например:

Если МонопольныйРежим() Тогда
Возврат;
КонецЕсли;

ФоновыеЗадания.Выполнить("...")

5. В некоторых случаях возникает необходимость в выполнении длительных операций, требующих установки монопольного режима доступа к информационной базе. Например:

- Обновление данных ИБ при первом интерактивном запуске программы после обновления конфигурации;
- Удаление объектов, помеченных на удаление;
- Выгрузка данных информационной базы в файл для перехода в сервис;
- Использования монопольного режима для снижения времени выполнения массовых операций по изменению данных;

При этом необходимо сначала устанавливать монопольный режим, а затем выполнять запуск фонового задания, которое реализует саму длительную операцию. В этом случае фоновым заданием будет унаследован монопольный режим, ранее установленный из пользовательского сеанса (см. [документацию к платформе](#)).

На время выполнения этого фонового задания следует блокировать весь интерфейс приложения, открывая форму ожидания завершения операции в режиме **РежимОткрытияОкна = БлокироватьВесьИнтерфейс**. Блокировать интерфейс приложения требуется потому, что на время выполнения задания полноценная работа пользователя с приложением уже невозможна:

- Если пользователь(*) попытается записать какой-либо объект, это приведет к ошибке (из-за установленного монопольного режима);
- В ряде случаев могут запускаться фоновые задания в качестве реакции на действия пользователя в случае (при поиске в динамическом списке, при вводе по строке, формировании отчетов и пр.), которые также завершаются с ошибкой.

Кроме того, на самой форме ожидания длительной операции не следует размещать элементы управления, которые могут приводить к запуску таких фоновых заданий. Например: поля ввода, динамические списки и отчеты.

* Примечание: ошибки записи также возникают в тех случаях, когда объекты записываются программно, например, из обработчиков ожидания. В них также следует проверять монопольный режим согласно п.5.

См. также

- [Длительные операции на клиенте](#)

Длительные операции на клиенте

Область применения: управляемое приложение.

#std755

Следует избегать длительного выполнения клиентского кода, помимо длительных вызовов серверного кода из клиентского (см. «[Длительные операции на сервере](#)»), т.к. это приводит к очутимым задержкам при работе с программой или даже к зависанию. Не следует выполнять потенциально длительные операции (такие как: обращение к сетевым ресурсам, «тяжелые» алгоритмы обработки данных на клиенте) в обработчиках ожидания и обработчиках событий элементов форм.

Длительные клиентские операции допустимо выполнять, только когда пользователь инициирует их явным образом (например, нажатием на кнопку).

Пример 1

Неправильно:

В обработчике ожидания обращаться к веб сервису для получения информации об обновлениях.

Правильно:

Получать информацию об обновлениях только при нажатии на кнопку, либо перенести получение информации об обновлениях на сервер:

- В регламентном задании получать информацию об обновлениях через веб сервис и сохранять полученную информацию, например, в регистре;
- В обработчике ожидания зачитывать уже полученную информацию об обновлении при помощи легкого запроса к регистру.

Пример 2

Неправильно:

Проверять доступность сетевого ресурса в обработчике события ПриИзменении поля формы, в котором вводится путь к этому сетевому ресурсу.

Правильно:

Проверять доступность сетевого ресурса в обработчике команды, которая выведена в интерфейс, например, в виде кнопки Проверить.

Формирование печатных форм

Область применения: управляемое приложение.

#std548

Некоторые справочники, документы и др. объекты конфигурации могут предоставлять команды по выводу их на печать. В данной статье приведены требования к реализации таких команд.

1.1. Для формирования печатной формы пользователю должно быть достаточно прав:

- на чтение основных объектов метаданных, по данным которых формируется печатная форма (например, для формирования печатной формы документа **ЗаказПокупателя** нужны права на чтение этого документа);
- и на выполнение команды, которая инициирует формирование печатной формы (если у пользователя есть права на документ **ЗаказПокупателя**, то в общем случае, это не приводит к тому, что ему автоматически становятся доступны все печатные формы; права на печатные формы могут быть ограничены отдельно с помощью прав на команды).

1.2. Если в печатную форму дополнительно выводятся данные каких-либо других объектов, связанных с основными, то права на них не должны оказывать влияния на возможность формирования печатной формы, а также на состав выводимой на печать информации. Поэтому код формирования печатной формы следует выполнять в таком случае в привилегированном режиме.

Например, для формирования печатной формы «**Счет-фактура (в валюте)**» используются данные документа, на основе которого формируется счет-фактура, и данные регистра сведений «**Курсы валют**». Пользователь может сформировать печатную форму, если у него есть права на чтение документа и на выполнение команды печати. Но при этом не имеет значения, есть ли у него права на регистр сведений «**Курсы валют**».

См. также: [Использование привилегированного режима](#)

2. Если функция печати предполагает множественную печать нескольких объектов, то по соображениям производительности (снижение нагрузки на СУБД) необходимо выполнять выборку данных в одном запросе, результаты которого затем обходятся в цикле.

См. также: [Многократное выполнение однотипных запросов](#)

Иключение из этого правила могут составлять случаи, когда выборка данных для нескольких объектов в одном запросе

- заметно усложняет разработку самого запроса
- наоборот, может привести к деградации производительности. Например, когда запрос содержит обращение к виртуальным таблицам, формируемым на определенную дату (как в случае с виртуальными таблицами остатков регистра накопления, срезом последних периодического регистра сведений и т.п.).

3.1. Печатная и экранная формы объектов (справочников, документов и др.), в которых выводятся табличные части, должны соответствовать друг другу по составу и порядку строк табличных частей. В частности, при выводе на печать не следует каким-либо образом дополнительно группировать строки табличных частей.

3.2. Однако, если строки в экранной форме уже сгруппированы по каким-либо признакам, то при выводе на печать следует применять такую же группировку.

Например, рассмотрим печатную и экранную формы документа, в которых выводится табличная часть цен номенклатуры по типам цен: для каждой номенклатуры в колонках выводятся розничная, оптовая и другие типы цен, но при этом «физически» одна строка табличной части документа содержит информацию о цене только одного типа.

4.1. В табличных частях печатных форм следует всегда выводить колонку со значением реквизита «Номер строки». В этом случае пользователи всегда смогут легко сопоставить строки табличных частей в печатных и экранных формах.

4.2. В тех случаях, когда на экранной форме не отображается реквизит «Номер строки», строки в печатной форме нумеруются в порядке их вывода.

4.3. Заголовок колонки с номером строки зависит от прикладной специфики печатной формы и может различаться в разных печатных формах. Например, в ТОРГ-12 колонка имеет строго регламентированное название «Номер по порядку», а в нерегламентированных печатных формах – может называться «№».

5. Строки табличных частей следует выводить в печатных формах отсортированными по полю «Номер строки», поскольку порядок вывода может быть разным на различных СУБД.

Исключение могут составлять печатные формы, прикладная специфика которых требует сортировки по другому реквизиту. Например, в задании кладовщику на отбор товаров из складских ячеек, строки задания отсортированы по порядку обхода ячеек склада (при этом значение реквизита «Номер строки» также выводится в соответствующей колонке).

См. также: [Упорядочивание результатов запроса](#)

6.1. При выводе данных в печатные формы, необходимо обеспечить, чтобы они были выведены полностью и не обрезались.

Методическая рекомендация (полезный совет)

6.2. В случае если печатные формы формируются с помощью табличных макетов, то при выводе строк табличных частей рекомендуется устанавливать свойство **АвтоВысотаСтроки** (объекта **ОбластьЯчеекТабличногоДокумента**) в значение **Истина**. Это позволяет избежать обрезания длинных текстовых строк, когда ширины колонок таблиц недостаточно.

7. При подготовке табличного документа на сервере не следует устанавливать размер полей заведомо меньше физических ограничений полей у принтера. В частности, недопустимо устанавливать нулевые поля.

В общем случае, размер полей табличного документа рекомендуется не менять (оставлять по умолчанию), кроме тех случаев, когда они явно регламентированы или стандартизированы. Например, согласно ГОСТ Р 6.30-2003 "Унифицированные системы документации. Унифицированная система организационно-распорядительной документации. Требования к оформлению документов" в документе должны быть заданы поля: 20 мм левое, 10 мм правое, 20 мм верхнее, 20 мм нижнее.

Требование обусловлено тем, что получение физических ограничений полей клиентского принтера не представляется возможным на сервере приложений, в связи с чем, при печати документа с нулевыми полями текст, выступающий за минимально допустимые поля принтера, не будет напечатан.

Пример некорректного кода:

```
// Зададим параметры печатной формы по умолчанию
Табдокумент.ПолеСверху      = 0;
Табдокумент.Полеслева        = 0;
Табдокумент.Полеснизу        = 0;
Табдокумент.Полесправа      = 0;
```

8. Если в конфигурации предусмотрена возможность того, что макет печатной формы может быть отредактирован пользователем в режиме предприятия (например, средствами подсистемы «Печать» Библиотеки стандартных подсистем), то необходимо рассчитывать на то, что любые параметры в нем могут быть изменены или удалены. Поэтому для повышения устойчивости кода формирования печатной формы следует избегать явного присвоения значений параметров в областях печати. Вместо этого следует использовать глобальный метод **ЗаполнитьЗначенияСвойств** или метод **Заполнить** коллекции **ПараметрыМакетаТекстовогоДокумента**.

Неправильно:

```
ОбластьПечати.Параметры.Организация = ДанныеПечати.Организация; // будет ошибка при отсутствии в макете параметра Организация
ОбластьПечати.Параметры.Контрагент = ДанныеПечати.Контрагент;
```

Правильно:

```
ЗаполнитьЗначенияСвойств(ОбластьПечати.Параметры, ДанныеПечати);
```

или

```
ОбластьПечати.Параметры.Заполнить(ДанныеПечати);
```

Примечание: в переменной **ДанныеПечати** в примерах может быть **Структура**, **Соответствие**, **ВыборкаИзРезультатаЗапроса** или любая другая коллекция со значениями параметров.

См. также

- [Названия печатных форм учетных документов и команд по их выводу на печать](#)

Открытие форм

Область применения: управляемое приложение, мобильное приложение.

#std404

1. Для открытия форм следует применять метод глобального контекста **ОткрытьФорму** (при использовании версии платформы 1С:Предприятие 8.2 и более ранних версий - также **ОткрытьФормуМодально**). Применение альтернативного способа, с получением формы и ее последующим открытием с помощью метода **ПолучитьФорму**, не рекомендуется.

Рекомендация обусловлена соображениями

- повышения устойчивости кода, работающего с формой, за счет разделения программного интерфейса для работы с формой и деталей ее внутренней реализации,
- а также сохранения единой стилистики кода прикладных решений.

Кроме того, применение глобального метода **ОткрытьФорму** гарантирует выполнение инициализации формы на сервере в обработчике **ПриСозданииНаСервере**. Этот подход помогает сосредоточить весь код инициализации формы в одном месте и исключает "случайное" обращение к серверу, связанное с инициализацией формы, между строками кода

```
Форма = ПолучитьФорму(...)
```

и

```
Форма.ОткрытьФорму(...)
```

См. также: раздел «Открытие управляемой формы» статьи [«Минимизация количества серверных вызовов»](#)

2. В случаях когда форма требует параметризации при открытии, все ее параметры следует указывать в наборе параметров формы. Таким образом, набор параметров формы декларативно описывает возможности формы по ее параметризации.

Параметры формы из этого набора могут быть указаны в вызывающем коде при открытии формы (**ОткрытьФорму**).

3. Не следует применять другие способы параметризации формы при открытии. Например, нужно избегать обращения к методам и свойствам формы после ее открытия. Например, вместо

```
МояФорма = Форма.ОткрытьФорму("ОбщаяФорма.ПутеводительПоСистеме");
МояФорма.Элементы.Группашаг.ТекущаяСтраница = МойФорма.Элементы.Группашаг.Страницы.Приветствие;
```

следует по той же причине использовать параметры формы:

```
ОткрытьФорму("ОбщаяФорма.ПутеводительПоСистеме", Новый Структура("РежимОткрытия", "Приветствие"));
```

4. Для получения результата работы формы, вместо непосредственного обращения к элементам и реквизитам формы

```
ФормаВопроса = ПолучитьФорму("ОбщаяФорма.ФормаВопроса");
ФормаВопроса.ОткрытьМодально();
Если ФормаВопроса.БольшеНеПоказыватьНапоминание Тогда
// ...
```

следует использовать процедуры-обработчики оповещений, которые будут вызваны при завершении работы пользователя с формой:

Оповещение = Новый ОписаниеОповещения("БольшеНеПоказыватьНапоминаниеЗавершение", ЭтотОбъект);
ОткрытьФорму("ОбщаяФорма.ФормаВопроса",,,, Оповещение, РежимОткрытияОкнаФормы.БлокироватьВесьИнтерфейс);
...

&НаКлиенте

Процедура БольшеНеПоказыватьНапоминаниеЗавершение(БольшеНоНапоминание, Параметры) Экспорт

```
Если БольшеНоНапоминание = Неопределено Тогда  
    Возврат;  
КонецЕсли;
```

```
Если БольшеНоНапоминание Тогда  
// ...
```

КонецПроцедуры

При этом возвращаемое значение формы формируется в коде модуля формы с помощью метода формы **Закрыть**.

См. также: [Ограничения на использование экспортных процедур и функций](#)

5. Другие ограничения:

- Обработчик события формы **ПриОткрытии** не должен содержать код по открытию какой-либо другой формы, так как это может привести к нарушению порядка отображения окон. В этом случае рекомендуется использовать обработчик ожидания на короткий интервал или открывать другие формы интерактивно, например, по нажатию на кнопку.
- Не рекомендуется выполнять программное открытие и закрытие формы в одном обработчике. Такие действия должны быть разнесены по времени. Например, закрытие формы можно выполнять в обработчике ожидания.
- При использовании в конфигурации **Библиотека стандартных подсистем** и разработке форм (рабочих мест), предназначенных только для **внешних пользователей**, следует явно блокировать открытие таких форм в сеансах "обычных" пользователей. Для этого следует устанавливать параметр **Отказ** при создании формы на сервере с помощью функции **ЭтоСеансВнешнегоПользователя** общего модуля **Пользователи** или **ПользователиКлиент**:

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

```
Если Не ПользователиКлиентСервер.ЭтоСеансВнешнегоПользователя() Тогда  
    Отказ = Истина;  
    Возврат;  
КонецЕсли;  
...
```

КонецПроцедуры

6. Следующие виды форм должны быть всегда доступны пользователю в режиме **1С:Предприятия** из меню "Все функции" вне зависимости от того, размещены ли соответствующие объекты в командном интерфейсе приложения или нет:

- основная форма списка (для всех объектов)
- основная форма обработки
- основная форма отчета

См. также

- [Открытие параметризованных форм](#)

Открытие параметризованных форм

Область применения: управляемое приложение, мобильное приложение.

#std741

1. В случаях когда форма требует параметризации при открытии, предназначена для открытия только при помощи встроенного языка и, как следствие, не может быть открыта из пункта меню "Все функции", не следует назначать такую форму основной формой объекта.

2. Если же у объекта нет других форм, которые могли бы быть назначены основными, то следует сделать основной эту параметризованную форму. В обработчике **ПриСозданииНаСервере** модуля формы необходимо проверять параметры формы и, если они не заполнены, вызывать исключение. Текст исключения должен указывать пользователю причину, по которой форма не может быть открыта.

Правила создания модулей форм

Область применения: управляемое приложение, мобильное приложение.

#std630

Область применения (уточнение): управляемое приложение, обычное приложение.

1. В модуле формы размещаются процедуры и функции, которые необходимы только для реализации логики работы этой формы и исполняются в контексте этой формы.

модули объектов, модули менеджеров объектов или общие модули. Для передачи входных параметров в форму следует применять метод **ОткрытьФорму**, для получения результата - процедуры-обработчики оповещений (см. стандарт [Открытие форм](#)), а для взаимодействия с открытыми формами - метод **Оповестить** и обработчик события **ОбработкаОповещения** (см. стандарт [Обновление списков при интерактивных действиях пользователя](#)).

1.2. Неправильно использовать экспортные процедуры и функции формы для параметризации формы при открытии. Например, неправильно:

```
Форма = ПолучитьФорму("ОбщаяФорма.МояФорма");  
Форма.Открыть();  
Форма.УстановитьПараметрСПомощьюЭтоЕкспортнойФункции(РежимРаботы);
```

правильно:

```
ПараметрыФормы = Новый Структура("РежимРаботы", РежимРаботы)  
ОткрытьФорму("ОбщаяФорма.МояФорма", ПараметрыФормы)
```

См. также: [Открытие форм](#)

1.3. Также не следует вызывать экспортные процедуры и функции модуля формы для обновления данных формы или для программной перерисовки формы в результате действий пользователя в других формах. В этом случае следует использовать метод глобального контекста **Оповестить**, методы формы **ОповеститьОЗаписиНового**, **ОповеститьОбАктивизации** и **ОповеститьОВыборе**. См. также: [Обновление списков при интерактивных действиях пользователя](#).

1.4. Исключения из этого правила составляют экспортные процедуры-обработчики оповещений (ОписаниеОповещения.ИмяПроцедуры).

2. В некоторых случаях в модуле формы возникает необходимость реализации процедур и функций, которые выполняются как на стороне клиента, так и на сервере. Например, для обновления данных формы из серверного обработчика **ПриСозданииНаСервере** и из клиентских событий **ПриИзменении**.

Для того чтобы избежать дублирования кода, такие процедуры и функции необходимо размещать в модуле формы с директивой компиляции **&НаКлиентеНаСервереБезКонтекста** и передавать в них контекст самостоятельно в виде параметра. В качестве контекста может выступать либо сама форма (**ФормаКлиентскогоПриложения**), либо ее реквизит (**ДанныеФормыСтруктура**).

Пример:

```
&НаСервере
Процедура ПриСозданииНаСервере()
    УстановитьДоступность(ЭтотОбъект);
КонецПроцедуры

&НаКлиенте
Процедура РазрешитьРедактированиеСуммыПриИзменении(Элемент)
    УстановитьДоступность(ЭтотОбъект);
КонецПроцедуры

&НаКлиентеНаСервереБезКонтекста
Процедура УстановитьДоступность(Форма)
    Форма.Элементы.ТоварыСумма.Доступность = Форма.Объект.РазрешитьРедактированиеСуммы;
КонецПроцедуры
```

См. также

- [Структура модуля](#)
- [Контекстная и внеоконтекстная передача управления на сервер](#)

Блокирующее или независимое открытие форм объектов

Область применения: управляемое приложение.

#std742

1. По умолчанию, при создании форм некоторых типов объектов (справочников, планов видов характеристик и др.) им устанавливается режим открытия с блокированием окна владельца (свойство **РежимОткрытияОкна** установлено в значение **БлокироватьОкноВладельца**). Для таких форм предполагается, что работа с ними всегда выполняется "за один заход", без необходимости переключения в другие формы. Тем самым, предотвращается одновременное открытие нескольких форм такого типа, и как следствие, рабочее пространство пользователя не "замусоривается" лишними окнами.

2. Это умолчание платформы рекомендуется изменять в тех случаях, когда форма представляет из себя «рабочее место», т.е. с помощью нее осуществляется доступ к большому функциональному блоку. В частности, если при работе с формой в обычных сценариях работы может потребоваться

- открытие других самостоятельных форм (отчетов, справочников и т.д.) из глобального командного интерфейса приложения;
- сравнение двух и более объектов между собой;

а также если в панели навигации формы

- содержится большое количество переходов к связанной информации (например, к подчиненным справочникам);
- есть переходы к параметризуемым отчетам.

Для таких форм необходимо установить свойству **РежимОткрытияОкна** значение **Независимый**. Как правило, это формы документов.

3. Для конфигураций, разрабатываемых на платформе 1С:Предприятие 8.3.9 и ранее, не следует открывать формы в режиме **ВариантОткрытияОкна.ОтдельноеОкно**, поскольку в конфигурациях, рассчитанных на работу в режимах «Такси» или «В закладках», такой режим открытия форм значительно усложняет пользователю настройку приложения для работы в веб-клиенте.

В большинстве случаев такие формы можно перевести на открытие в режиме блокирования окна владельца или независимо (что технически не приводит к открытию отдельного окна в веб-браузере в режимах интерфейса «Такси» или «В закладках», но визуально будет выглядеть для пользователя так же, как открытие отдельного окна).

Иключение составляют случаи, когда в конфигурации предусмотрено, что форма открывается таким способом только в тонком клиенте или только в режиме «В отдельных окнах».

См. также

- Выбор: блокирующая форма или независимая в руководстве по стилю.

Ограничение на использование модальных окон и синхронных вызовов

Область применения: управляемое приложение.

#std703

Действует для конфигураций, разрабатываемых на платформе 1С:Предприятие 8.3 и выше.

1. При разработке конфигураций, предназначенных для работы в веб-клиенте, запрещено использовать модальные формы и диалоги. В противном случае, конфигурация окажется неработоспособной в ряде веб-браузеров, так как модальные окна не входят в стандарт веб-разработки. Для разработки качественных веб-приложений требуются асинхронные средства обеспечения взаимодействия с пользователем, которые предоставляет платформа 1С:Предприятие.

2. Для этого свойство конфигурации **Режим использования модальности** должен быть установлено в **Не использовать**, а вместо модальных методов следует вызывать их немодальные аналоги с блокированием окна владельца или всего интерфейса.

3. В процедуре **ПриЗавершенииРаботыСистемы** модуля управляемого приложения недопустимо использовать асинхронные вызовы.

4. Если в процедуре **ПередЗавершениемРаботыСистемы** модуля управляемого приложения используются асинхронные вызовы, то в ней необходимо установить значение параметра **Отказ = Истина** и из процедуры оповещения о завершении асинхронного вызова продолжить завершение работы системы.

Пример:

```
Процедура ПередЗавершениемРаботыСистемы(Отказ)
    ДополнительныеПараметры = Новый Структура;
    ДополнительныеПараметры.Вставить("ЗавершитьРаботуСистемы", Истина);
    ОписаниеОповещения = Новый ОписаниеОповещения("ПослеУдаленияФайлов", РаботаСФайламиКлиент, ДополнительныеПараметры);
    НачатьУдалениеФайлов(ОписаниеОповещения, ПутьКФайлу);
    Отказ = Истина;
КонецПроцедуры
```

// Общий модуль РаботаСФайламиКлиент:

```
Процедура ПослеУдаленияФайлов(ДополнительныеПараметры) Экспорт
    Если ДополнительныеПараметры.ЗавершитьРаботуСистемы Тогда
        // Больше нет действий перед завершением работы системы.
        ЗавершитьРаботуСистемы();
    КонецЕсли;
КонецПроцедуры
```

5. При переработке синхронных вызовов на их асинхронные аналоги можно включать флагок **Поиск использования синхронных вызовов при проверке конфигурации** (конфигуратор – меню **Конфигурация – Проверка конфигурации...**). Но при этом из результатов проверки потребуется вручную исключать все корректные места вызовов в коде, который не исполняется в веб-клиенте (например, серверный код).

См. также

- [Отказ от использования модальных окон](#) (статья на ИТС)
- [Перевод конфигураций на платформе "1С:Предприятие 8.2" на платформу "1С:Предприятие 8.3" без режима совместимости с версией 8.2](#) (статья на ИТС)

- [Общие требования к конфигурации](#)

Запрет редактирования полей таблицы по условию

Область применения: управляемое приложение.

#std399

Методическая рекомендация (полезный совет)

В некоторых случаях требуется предупредить пользователя о том, что ввод тех или иных данных в таблице управляемой формы не имеет смысла или вообще не может быть выполнен до заполнения других полей таблицы. В таких случаях рекомендуется применять описанный ниже прием.

- Для предупреждения пользователя применять свойства поля **ОтображениеПредупрежденияПриРедактировании** и **ПредупреждениеПриРедактировании**
- Текст предупреждения указать в свойстве поля формы при разработке формы в конфигураторе.
- Управлять свойством **ОтображениеПредупрежденияПриРедактировании** из встроенного языка при активизации строки таблицы и при изменении тех значений, которые влияют на условие редактирования поля
- Дать пользователю возможность редактирования уже введенное значение, даже если оно введено "не по правилам", для того, чтобы пользователь мог удалить ранее введенное значение.

Например, в таблице формы **Затраты** имеются поля **СтатьяЗатрат** и **АналитикаСтатьяЗатрат**. При этом заполнение поля **АналитикаСтатьяЗатрат** не имеет смысла без заполнения поля **СтатьяЗатрат**. Тогда код формы может включать примерно следующие процедуры.

```
&НаКлиенте
Процедура УстановитьПредупреждениеПриРедактировании()
    Если НЕ ЗначениеЗаполнено(Элементы.Затраты.ТекущиеДанные.СтатьяЗатрат) Тогда
        Если НЕ ЗначениеЗаполнено(Элементы.Затраты.ТекущиеДанные.АналитикаСтатьяЗатрат) Тогда
            Элементы.ЗатратыАналитикаСтатьяЗатрат.ОтображениеПредупрежденияПриРедактировании =
                ОтображениеПредупрежденияПриРедактировании.Отображать;
        Иначе
            Элементы.ЗатратыАналитикаСтатьяЗатрат.ОтображениеПредупрежденияПриРедактировании =
                ОтображениеПредупрежденияПриРедактировании.НеОтображать;
        КонецЕсли;
    Иначе
        Элементы.ЗатратыАналитикаСтатьяЗатрат.ОтображениеПредупрежденияПриРедактировании =
            ОтображениеПредупрежденияПриРедактировании.НеОтображать;
    КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ЗатратыПриАктивизацииСтроки(Элемент)
    УстановитьПредупреждениеПриРедактировании();
КонецПроцедуры

&НаКлиенте
Процедура ЗатратыСтатьяЗатратПриИзменении(Элемент)
    УстановитьПредупреждениеПриРедактировании()
КонецПроцедуры
```

В том случае если от некоторых условий зависит не только сама необходимость предупреждения, но и текст самого предупреждения, следует управлять из встроенного языка не только свойством поля **ОтображениеПредупрежденияПриРедактировании**, но и свойством **ПредупреждениеПриРедактировании**.

Особенности табличного документа в веб-клиенте

Область применения: управляемое приложение.

#std533

Методическая рекомендация (полезный совет)

1. Для достижения приемлемой скорости работы форм в веб-клиенте, в которых пользователь редактирует табличный документ, необходимо придерживаться следующих рекомендаций.

1.1. В случае если после изменения пользователем значения ячейки требуется выполнить перерасчет состояния формы (установить значения соседних зависимых ячеек, доступность элементов управления и т.п.), следует избегать обращений на сервер, а выполнять весь пересчет на клиенте. Контекстный вызов на сервер для формы с табличным документом может сделать работу пользователя в форме неприемлемой.

При этом важно, чтобы клиентская логика обновления была оптимизирована и отрабатывала быстро. В частности:

- вместо полного пересчета всех ячеек документа следует «точечно» обновлять только те соседние ячейки, значения которых зависят от измененной ячейки;
- не обновлять динамически состояние кнопок в зависимости от текущей области табличного документа; а например, сообщать пользователю при нажатии на кнопку, почему действие в данный момент недоступно, либо автоматически определять ближайшую подходящую область документа для выполнения действия.

1.2. В тех случаях, когда серверный вызов при завершении редактирования ячейки все же необходим, следует убедиться, что вызов сервера выполняется только один. По возможности, рекомендуется использовать бесконтекстные вызовы (&НаСервереБезКонтекста).

См. также: [Минимизация клиент-серверного взаимодействия](#).

1.3. Следует избегать в макетах табличных документов большого количества колонок и объединений. Для этого рекомендуется использовать переменную ширину колонок для различных строк. При этом внешний вид документа для пользователя не меняется.

Это упрощение позволяет снизить время открытия формы, а также интервал ожидания пользователя при выполнении действий в форме, предполагающих контекстный вызов сервера.

2. В веб-браузерах Microsoft Internet Explorer 6.0 и 7 не поддерживается отображение объединенных ячеек табличного документа, у которых указана рамка слева и не указана рамка снизу. Для таких ячеек рекомендуется указывать рамку снизу.

3. В веб-клиенте, если у ячейки табличного документа установлено выравнивание по правому краю и текст не полностью помещается в ячейке, то помещающаяся часть текста выравнивается по левому краю, при этом текст обрезается. Поэтому если отображение только части содержимого ячейки не имеет смысла (как правило, это так для числовых полей - нет смысла показывать половину числа), то ячейке следует устанавливать свойство Размещение в значение **Забивать**. Если же урезанное значение имеет смысл (строковые наименования и т. д.), то следует устанавливать значение **Обрезать**.

Обращение из кода к автоматически формируемым элементам управления формы

Область применения: управляемое приложение.

#std536

Недопустимо управлять из кода элементами управления формы, которые автоматически формируются платформой.

Например:

- неправильно при открытии формы из кода прятать команды группы **Создать на основании** в зависимости от тех или иных условий.
- правильно: использовать [обработчик события ОбработкаЗаполнения](#).

См. также

- [Обращение из кода к пользовательским элементам управления формы](#)

Обращение из кода к пользовательским элементам управления формы

Область применения: управляемое приложение.

#std545

Методическая рекомендация (полезный совет)

При разработке логики работы формы следует иметь в виду, что если пользователь в режиме **1С:Предприятия** добавляет какие-либо элементы формы (**Все действия - Изменить форму**), то обращение к таким элементам из встроенного языка невозможно.

Например, если в элемент формы **Группа** вида **Страницы** пользователь добавил страницу, то при переходе на эту страницу, свойство группы **ТекущаяСтраница** станет равно **Неопределено**. Свойство формы **ТекущийЭлемент** также будет содержать значение **Неопределено** в том случае, когда активным элементом формы является элемент, добавленный пользователем.

Например, вместо непосредственного обращения к свойствам элементов группы со страницами:

ИмяТекущейСтраницы = Элементы.ГруппаСоСтраницами.ТекущаяСтраница.Имя;

следует предварительно выполнять проверку на **Неопределено**:

ТекущаяСтраница = Элементы.ГруппаСоСтраницами.ТекущаяСтраница;
ИмяТекущейСтраницы = ?(ТекущаяСтраница <> Неопределено, ТекущаяСтраница.Имя, "");

См. также

- [Обращение из кода к автоматически формируемым элементам управления формы](#)

Команды по модификации объектов

Область применения: управляемое приложение.

#std537

Свойство **Изменяет данные** должно быть установлено в **Истина** для всех команд, которые изменяют или могут изменять данные объекта.

Это правило должно соблюдаться и для тех команд, которые в некоторых сценариях работы могут и не изменить данные. Например, если на форме имеется команда по заполнению табличной части документа, которая перед выполнением задает пользователю вопрос вида: «Перед выполнением операции табличная часть будет очищена. Продолжить?», то в случае отказа пользователя, выполнение команды прерывается, и изменения табличной части не происходит. Тем не менее, если пользователь ответит утвердительно, то табличная часть будет изменена. Поэтому в целом для этой команды должен быть установлен признак **Изменяет данные**.

При несоблюдении этого правила:

- не будет выполняться установка свойства **Только просмотр** для данной команды при установке в **Истина** этого свойства для формы в целом.
- при выполнении команды не произойдет попытки автоматической блокировки объекта для редактирования, что может привести к невозможности его последующего сохранения, в том случае, если в это же время другой пользователь изменит данный объект.

Контекстная и внеконтекстная передача управления на сервер

Область применения: управляемое приложение, мобильное приложение.

#std636

1. Платформа **1С:Предприятие** позволяет передавать управление из клиентского в серверный код модуля формы двумя способами: контекстно и внеконтекстно.

При внеконтекстной передаче управления на сервер передаются только те данные, которые явно специфицированы разработчиком в параметрах процедуры (функции) с директивой компиляции **&НаСервереБезКонтекста**.

При контекстной передаче управления на сервер, помимо параметров процедуры (функции) с директивой компиляции **&НаСервере**, передаются еще и данные формы, которые были изменены на клиенте за период с момента предыдущего контекстного серверного вызова (см. ниже приложение). При этом на сервере выполняется ряд дополнительных действий по инициализации методов формы и серверной копии данных формы, что может увеличивать общее время, которое сервер затрачивает на обработку вызванной процедуры (функции).

2. Контекстную передачу управления следует использовать в случаях когда:

- платформа **1С:Предприятие** самостоятельно оптимизирует объем передаваемых данных между клиентом и сервером (в обоих направлениях). Прежде всего, это реквизиты формы с табличными документами и коллекции элементов (**ДанныеФормыКоллекция**, **ДанныеФормыСтруктураСКоллекцией**, **ДанныеФормыДерево**). См. также: [Использование объекта ДанныеФормыКоллекция](#).
- и при этом затраты ресурсов сервера на инициализацию контекста формы оправдываются существенным снижением трафика между клиентом и сервером и [снижением числа вызовов сервера](#).

В остальных случаях рекомендуется использовать внеконтекстную передачу управления с клиента на сервер.

3. При передаче управления с клиента на сервер недопустимо использовать объекты типов **ДанныеФормыСтруктура**, **ДанныеФормыКоллекция**, **ДанныеФормыСтруктураСКоллекцией**, **ДанныеФормыДерево** и **ТабличныйДокумент** в качестве параметров функции, передаваемых по значению. При передаче таких типов по значению с клиента на сервер всегда передается полная копия объекта, а не только измененные данные.

Вести работу с этими типами следует на сервере, для чего переходить с клиента на сервер с помощью явного контекстного вызова сервера.

Например, неправильно:

// Модуль формы

&НаКлиенте

Процедура КоличествоОтмененныхСтрочекЗаказа()

 КоличествоСтрочек = ОбщийМодульВызовСервера.КоличествоОтмененныхСтрочек(Объект.Товары); // Неоптимальная передача на сервер табличной части "Товары"

КонецПроцедуры

// Общий серверный модуль ОбщийМодульВызовСервера

Функция КоличествоОтмененныхСтрочек(ТабличнаяЧасть);

 НайденныеСтроки = ТабличнаяЧасть.НайтиСтроки(Новый Структура("Отменено", Истина));
 Возврат НайденныеСтроки.Количество();

КонецФункции

Правильно:

```
// Модуль формы

&НаКлиенте
Процедура КоличествоОтмененныхСтроЖаказа()
    КоличествоСтр = КоличествоОтмененныхСтроЖаказа(); // Передача табличной части "Товары" выполняется неявно платформой, оптимально
КонецПроцедуры

&НаСервере
Процедура КоличествоОтмененныхСтроЖаказа()
    Возврат ОбщийМодульВызовСервера.КоличествоОтмененныхСтроЖаказа(Объект.Товары); // вызов "сервер"-сервер" без доп. накладных расходов
КонецПроцедуры

// Общий серверный модуль ОбщийМодульВызовСервера

Функция КоличествоОтмененныхСтроЖаказа(ТабличнаяЧасть);
    НайденныеСтроки = ТабличнаяЧасть.НайтиСтроки(Новый Структура("Отменено", Истина));
    Возврат НайденныеСтроки.Количество()
КонецФункции
```

Приложение

При контекстной передаче управления на сервер в платформе **1С:Предприятие** действуют следующие правила передачи измененных данных формы между клиентом и сервером:

- значения реквизитов формы типа **ДанныеФормыСтруктура** передается целиком, в случае если изменился хотя бы один из его реквизитов;
- для объектов, представленных типами **ДанныеФормыКоллекция** (**ДанныеФормыСтруктураСКоллекцией**, **ДанныеФормыДерево**) изменения учитываются с "точностью" до каждого элемента коллекции – передаются только измененные элементы. При этом измененные элементы коллекций передаются целиком. См. также: [Использование объекта ДанныеФормыКоллекция](#).
- для объектов типа **ТабличныйДокумент** передаются только измененные области;
- объекты типа **ДинамическийСписок** не передаются.

См. также

- [Правила создания модулей форм](#)

Использование объекта ДанныеФормыКоллекция

Область применения: управляемое приложение, мобильное приложение.

#std628

1. В целях оптимизации объема данных, передаваемых между клиентом и сервером, платформа **1С:Предприятие** по-особому организует передачу объектов формы типа **ДанныеФормыКоллекция**. Данные таких объектов передаются определенными порциями таким образом, что новые порции данных передаются с сервера на клиент только по мере обращения к этим данным на клиенте. Необходимо учитывать эту особенность при разработке форм, т.к. в противном случае, код формы может приводить к излишним неявным серверным вызовам, инициируемым платформой.

2. При работе с объектами типа **ДанныеФормыКоллекция**, если предполагается, что объект типа **ДанныеФормыКоллекция** может содержать большое количество строк (нужно ориентироваться на количество от 20 строк), необходимо придерживаться следующих рекомендаций:

- обход строк такой коллекции необходимо производить на сервере;
- функцию **НайтиСтроки** вызывать только на сервере.

Например, если решается задача по проверке в объекте строк, удовлетворяющих некоторому условию, то неправильно:

```
&НаКлиенте
Процедура ПроверитьНаличиеСтроЖаказа()
    Если Объект.Товары.НайтиСтроки(Новый Структура("Количество", 0)).Количество() > 0 Тогда
        Предупреждение(НСтр("ru = 'Есть строки с нулевым количеством'"));
    КонецЕсли;
КонецПроцедуры
```

правильно выполнять один явный вызов сервера:

```
&НаСервере
Функция ЕстьСтрокиСНулевымКоличеством()
    Возврат Объект.Товары.НайтиСтроки(Новый Структура("Количество", 0)).Количество() > 0;
КонецФункции
```

```
&НаКлиенте
Процедура ПроверитьНаличиеСтроЖаказа()
    Если ЕстьСтрокиСНулевымКоличеством() Тогда
        Предупреждение(НСтр("ru = 'Есть строки с нулевым количеством'"));
    КонецЕсли;
КонецПроцедуры
```

См. также

- [Минимизация количества серверных вызовов](#)

Условное оформление в формах

Область применения: управляемое приложение.

#std710

Рекомендация (полезный совет)

1. Для настройки некоторых свойств элементов управления можно использовать условное оформление. Однако у этого механизма также есть ряд ограничений.

1.1. Не следует использовать условное оформление для скрытия в таблице строк целиком. Это существенно замедляет работу в веб-клиенте, а также приводит к некорректному отображению содержащего таблицы.

1.2. Если задача может быть функционально решена как с помощью условного оформления динамического списка, так и с помощью условного оформления формы, то следует выбрать первый вариант (условное оформление динамического списка). Это также несколько ускорит открытие формы.

2.1. Настройку условного оформления форм и динамических списков рекомендуется делать в коде формы. Такой подход имеет ряд преимуществ перед заданием настроек условного оформления в свойствах формы:

- настройки однотипного условного оформления можно вынести в общие модули. Например, есть 80 форм, имеющих условное оформление: "если НЕ ХарактеристикиИспользуются, то в поле "Характеристика" вывести текст <характеристики не используются>", то можно вынести эту настройку в код процедуры общего модуля;
- при объединении конфигураций есть возможность объединять условное оформление (особенно это актуально при [разветвленной разработке конфигураций](#));
- при изменении в метаданных (например, перенименовании значения перечисления) условное оформление может перестать работать. Если условное оформление настраивается в коде конфигурации, то при синтаксическом контроле модулей эта ошибка будет выявлена. Так ошибки в настройках условного оформления будут выявляться средствами автоматизированной проверки (например, АПК), т.к. будет диагностироваться ошибка при попытке получения формы.

2.2. Все настройки условного оформления должны производиться при создании формы и потом не должны модифицироваться. Исключением могут являться случаи, когда элементы формы генерируются программно – условное оформление таких элементов нужно настраивать при генерации элементов и потом не нужно менять.

2.3. В коде процедуры установки условного оформления нужно минимизировать использование строковых констант, а использовать переменные, разыменования и т.д. – такой подход позволит минимизировать количество скрытых ошибок в настройках условного оформления, например:

```
&НаСервере
Процедура УстановитьУсловноеОформление()
    УсловноеОформление.Элементы.Очистить();
    Элемент = УсловноеОформление.Элементы.Добавить();
    ПолеЭлемента = Элемент.Поля.Элементы.Добавить();
    ПолеЭлемента.Поле = Новый ПолеКомпоновкиДанных(Элементы.ТоварыУпаковка.Имя);
    ГруппаОтбора1 = Элемент.Отбор.Элементы.Добавить(Тип("ГруппаELEMENTовОтбораКомпоновкиДанных"));
    ГруппаОтбора1.ТипГруппы = ТипГруппыELEMENTовОтбораКомпоновкиДанных.ГруппаИли;
    ОтборЭлемента = ГруппаОтбора1.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
    ОтборЭлемента.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("АдресноеХранение");
    ОтборЭлемента.ВидСравнения = ВидСравненияКомпоновкиДанных.Равно;
    ОтборЭлемента.ПравоеЗначение = Ложь;
    ОтборЭлемента = ГруппаОтбора1.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
    ОтборЭлемента.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Объект.Товары.ТипНоменклатуры");
    ОтборЭлемента.ВидСравнения = ВидСравненияКомпоновкиДанных.Неравно;
    ОтборЭлемента.ПравоеЗначение = Перечисления.ТипыНоменклатуры.Товар;
    ОтборЭлемента = ГруппаОтбора1.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
    ОтборЭлемента.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Объект.Статус");
    ОтборЭлемента.ВидСравнения = ВидСравненияКомпоновкиДанных.Равно;
    ОтборЭлемента.ПравоеЗначение = Перечисления.СтатусыПриходныхОрдеров.КПоступлению;
    Элемент.Оформление.УстановитьЗначениеПараметра("ОтметкаНезаполненного", Ложь);
КонецПроцедуры
```

См. также

- [Поле "Дата" в списках](#)
- [Правила создания модулей форм](#)

Ограничение использования поля HTML документа

Область применения: управляемое приложение.

#std730

1. Не следует использовать поля с HTML-документами (свойство **ВидПоля** установлено в **ВидПоляФормы.ПолеHTMLДокумента**) в случаях, когда возможно использование элементов управления платформы 1С:Предприятие. При разработке форм следует стремиться применять только штатные элементы управления, предусмотренные в платформе.

Например, неправильно, размещать на форме отдельные гиперссылки с помощью полей с HTML-документами.
Правильно использовать для этих целей кнопки, надписи с гиперссылками или с форматированными строками.

2. Допустимый пример использования поля с HTML-документом: вывод в формах различных пользовательских инструкций, встроенной справки, путеводителей с оформлением и картинками, которые предназначены только для просмотра информации.

При этом должно быть самостоятельно обеспечено корректное отображения страниц во всех видах клиентских приложений и на всех веб-браузерах, которые поддерживаются платформой 1С:Предприятие.

См. также

- [Особенности разработки конфигураций для ОС Linux](#)

Использование режима вертикальной прокрутки форм

Область применения: управляемое приложение.

#std734

Действует для конфигураций, разрабатываемых на платформе 1С:Предприятие 8.3 и выше.

1. Для конфигураций с режимом совместимости «Версия 8.3.3» и выше свойство «Вертикальная прокрутка» формы должно, как правило, иметь значение «Авто».

2. Исключением являются формы, имеющие поля для вывода HTML-документов, форматированных документов, текстовых документов, а также формы, кроме отчетов, имеющие поля для вывода табличных документов. Для таких форм свойство «Вертикальная прокрутка» формы должно быть установлено в значение «Использовать при необходимости».

Если форма была создана в режиме совместимости «Версия 8.3.2» или ниже – после включения режима совместимости «Версия 8.3.3» и выше свойство «Вертикальная прокрутка» автоматически устанавливается в значение «Использовать при необходимости». Правильное значение свойства для таких форм следует установить после смены режима совместимости.

См. также

- [Документация по платформе 8.3. Руководство разработчика. 7.5.2. Форма](#)

История выбора при вводе

Область применения: управляемое приложение.

#std744

1. Для свойства **История выбора** большинства объектов метаданных должно быть установлено значение **Авто**.

2.1. Историю выбора в свойствах объекта метаданных рекомендуется отключать, если ее использование не соответствует прикладной логике конфигурации:

- для объектов, сценарий использования которых не предполагает повторный выбор из 5 ранее выбранных вариантов.

Примеры:

Специфика использования большинства документов такова, что повторный их выбор маловероятен, например, выбор объекта расчетов в **Поступлении безналичных денежных средств**.

- для объектов, в модуле менеджера которых переопределена обработка получения данных выбора (есть обработчик **ОбработкаПолученияДанныхВыбора**), т.к. прописанные там условия не учитываются механизмом составления списка истории выбора. Поэтому, используя историю выбора в этом случае, пользователь может получить возможность выбрать значение, которое он не мог бы выбрать другими способами.

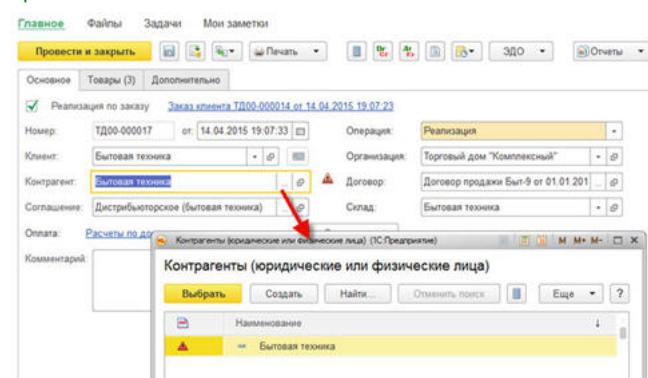
2.2. После отключения истории выбора в свойствах объекта метаданных необходимо, во всех ссылающихся на него полях ввода, установить для следующих свойств указанные ниже значения:

- КнопкаВыпадающегоСписка – Нет
- КнопкаВыбора – Да
- ОтображениеКнопкиВыбора – В поле ввода

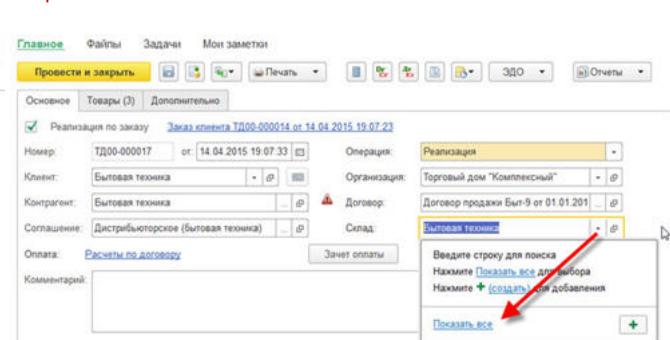
Это необходимо сделать, чтобы пользователю перед началом выбора в поле ввода не отображалось меню, в котором нужно всегда нажимать «Показать все».

Например:

Правильно



Неправильно



Исключения:

Можно не изменять значения свойств полей ввода если:

- для поля ввода установлен режим выбора из списка и заполнен (или в метаданных или программно) список выбора
- поле ввода ссылается на объект метаданных с установленным свойством **Быстрый выбор**

Для автоматического изменения свойств полей выбора можно воспользоваться приложенной [обработкой](#).

Ограничения при использовании динамических списков

Область применения: управляемое приложение, мобильное приложение.

#std489

1.1. При разработке интерфейса, разработчик может использовать группировки в динамических списках (см. [Группировки в списках](#)). Разработчик может:

- установить в настройках динамического списка группировки по умолчанию;
- добавить на форму специальные элементы управления (команды меню, поля выбора для «быстрой» группировки и т.п.), которые предоставляют пользователю возможность устанавливать группировки.

В данном стандарте перечислены условия, которые должен соблюсти разработчик, если он тем или иным способом управляет группировками динамических списков.

1.2. Пользователь, при помощи настройки списка может установить свои группировки – в этом случае прикладной разработчик не может (и не должен) гарантировать оптимальную производительность.

1.3. Использование группировок рекомендуется в тех динамических списках, в которых заведомо небольшое число записей (не более нескольких сотен). Небольшое число записей может обеспечиваться отбором, примененным в запросе динамического списка или отбором, применяемым к динамическому списку в форме и действие которого пользователь не может отменить.

1.4. В динамических списках, которые отображают таблицы с большим количеством записей, группировка может осуществляться только по проиндексированным полям.

Требование связано со следующими особенностями работы динамического списка. Для построения группировки, динамическому списку необходимо выбрать все уникальные из таблицы базы данных значения поля, по которому строится группировка. Затем производится сортировка и вывод пользователю. Когда пользователь раскрывает значение группировки, динамический список выбирает все записи таблицы с заданным значением в поле.

1.5. Допускается делать многоуровневые группировки в динамических списках только при соблюдении следующих условий:

- поле, по которому осуществляется первая группировка, должно быть проиндексировано;
- поле, по которому осуществляется первая группировка, должно обладать хорошей селективностью (т.е. для каждому значению этого поля должно соответствовать небольшое количество записей в таблице базы данных);

Эти требования связаны с тем, что раскрытие пользователем последующих (после первой) группировок динамический список будет отрабатывать уже без использования индексов, по всем элементам, отобранным по первой группировке.

1.6. Не рекомендуется делать группировки по полям, которые являются характеристиками объекта метаданных. Это ограничение связано с тем, что при выводе характеристик делается ЛЕВОЕ СОЕДИНЕНИЕ с таблицей характеристик, поэтому запрос с отбором по одной характеристике будет не эффективным даже при наличии индекса (в любом случае будет сканирование по главной таблице).

2. Для иерархических списков не рекомендуется устанавливать свойство **НачальноеОтображениеДерева** в значение **РаскрыватьВсеУровни**, так как это приведет к критичному снижению скорости открытия больших списков. Следует использовать значения **НеРаскрывать** или **РаскрыватьВерхнийУровень**.

Дополнительную информацию об особенностях динамических списков можно получить в документации по платформе (см. [Динамический список](#))

3. [Строку поиска в командной панели динамического списка](#) допустимо отключать в тех случаях, когда с его помощью не выполняются основные сценарии поиска. Например, в списке номенклатуры важен поиск по части артикула, что не поддерживается механизмом поиска.

Это ограничение обусловлено тем, что почти всегда поиск (когда он использует технологию полнотекстового поиска) работает только с начала слов. При этом в версиях платформы 1С:Предприятие 8.3.7 и ранее у данного механизма имеется еще более ограничений (в частности, в поисковую выдачу не попадали элементы, которые еще не были проиндексированы и т.п.).

Для отключения строки поиска в командной панели необходимо свойства **ПоложениеСтрокиПоиска** и **ПоложениеУправленияПоиском** динамических списков установить в значение **Нет** (для форм, созданных в предыдущих версиях платформы, значение **Нет** уже установлено по умолчанию).

См. также: [Запросы в динамических списках](#)

Особенности реализации команд для форм списков

Область применения: управляемое приложение, мобильное приложение.

#std495

[Методическая рекомендация \(полезный совет\)](#)

При разработке команд непосредственно в формах динамических списков, логика команды должна учитывать возможность того, что пользователь может сгруппировать динамический список по одной из колонок. Если логика команды не предусматривает обработку выделенных строк группировки, то их необходимо пропускать. При этом рекомендуется выдавать сообщение, если в списке выделена только одна строка группировки:

```
&НаКлиенте
Процедура КомандаВФорме(Команда)
МассивОбъектов = Элементы.Список.ВыделенныеСтроки;
Если МассивОбъектов.Количество() = 1
    И ТипЗнач(МассивОбъектов [0]) <> <ОжидаемыйТип> Тогда
        Предупреждение(НСтр("ru = 'Команда не может быть выполнена для указанного объекта.'"));
        Возврат;
КонецЕсли;
Для Каждого ОбъектаЗСписка Из МассивОбъектов Цикл
    Если ТипЗнач(ОбъектаЗСписка) <> <ОжидаемыйТип> Тогда
        Продолжить;
    КонецЕсли;
    // Обработка объекта строки.
    //
КонецЦикла;
КонецПроцедуры
```

См. также

- [Обновление списков при интерактивных действиях пользователя](#)

Организация работы со списками данных с помощью общих команд

Область применения: управляемое приложение.

#std397

[Методическая рекомендация \(полезный совет\)](#)

1. В некоторых случаях возникает необходимость в разработке специализированных форм списков, которые должны использоваться в том или ином сценарии работы пользователя и команды которых должны быть расположены в разных разделах командного интерфейса (разных "рабочих местах" пользователей).

2. Для того чтобы обеспечить отсутствие дублирования кода форм списков, облегчить их поддержку и, вместе с тем, обеспечить их использование в разных разделах командного интерфейса программы, рекомендуется следующее решение:

- Разработать параметризованную форму списка объекта, которая при создании формы на сервере на основании переданных параметров обеспечит разные прикладные свойства: тот или иной состав колонок, те или иные отборы и порядок элементов, а также параметризумый заголовок формы.
- Назначить эту форму основной формой списка объекта. Открытие формы без параметра при этом должно выглядеть для пользователя как открытие основной формы списка.
- Создать соответствующее число общих команд, которые открывают такую форму с тем или иным параметром. При этом заголовок открываемой формы должен соответствовать названию команды.
- Разместить общие команды в командном интерфейсе программы (отнести к той или иной подсистеме, разместить в тех или иных формах и т.п.)
- Обеспечить раздельное сохранение тех настроек формы, которые специфичны для каждого рабочего места (например, значения полей с отборами). См. свойство формы **КлючНазначенияИспользования**.

Пример подобной параметризованной формы списка можно найти в Библиотеке стандартных подсистем в форме **Задача.ЗадачаИсполнителя.ФормаСписка**.

3. Если по каким-то причинам разработка всей функциональности в рамках одной формы не оправдана, то допустимо следующее решение:

- разработать отдельные общие формы конфигурации, предназначенные для работы со списком объектов;
- разработать соответствующее число общих команд, которые открывают ту или иную форму;
- и разместить их в командном интерфейсе.

4. Также возможно комбинирование двух подходов, когда некоторые специализированные формы списков одного и того же объекта получаются параметризацией форм, а некоторые - реализацией отдельных форм. При этом предпочтительным решением остается вариант создания одной параметризованной формы.

Обновление списков при интерактивных действиях пользователя

Область применения: управляемое приложение, мобильное приложение.

#std558

1. Платформа 1С:Предприятие по умолчанию обновляет содержимое динамических списков при выполнении пользователем стандартных команд по изменению элементов этого списка (**Записать**, **Провести**, **Пометить на удаление** и т.п.) и при добавлении новых элементов. Но в ряде ниже перечисленных случаев такое обновление следует предусмотреть явно в прикладном коде.

2. Команды, которые изменяют объекты и размещаются в форме списка, должны заботиться об обновлении списка объектов после своего выполнения. Например, после изменения одного объекта рекомендуется вызывать метод **ОповеститьОбИзменении**:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)
```

```
ОбъектСсылка = ПараметрКоманды;
// Меняем объект
// ...
ОповеститьОбизменении(ОбъектСсылка);
```

КонецПроцедуры

При изменении нескольких объектов целесообразно обновлять список однократно, в конце операции:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)
```

```
Для Каждого ОбъектСсылка Из ПараметрКоманды Цикл
// Меняем объект
// ...
КонецЦикла;
```

```
ОповеститьОбизменении(ТипЗнч(ПараметрКоманды[0]));
```

КонецПроцедуры

3. Также должно быть обеспечено обновление данных в динамическом списке, у которого не назначена основная таблица. Такие динамические списки могут отображать данные из нескольких таблиц.

3.1. Если данные добавляются или изменяются командой, расположенной в форме динамического списка, то необходимо обновить список явно после выполнения команды.

3.2. Если данные изменяются в какой-либо другой форме (например, в форме объекта), то необходимо организовать обновление списка через оповещение. Для этого в каждой форме объекта, данные которого выводятся в динамическом списке, следует реализовать обработчик события **ПоследЗаписи** с использованием метода **Оповестить**:

```
Оповестить ("Запись_<ИмяОбъекта>", ПараметрыОповещения, ОбъектСсылка);
```

где

- **Запись_<ИмяОбъекта>** - имя события, в котором <ИмяОбъекта> - имя объекта, как оно задано в метаданных. Например, для документа "Расходная накладная" имя события будет "Запись_РасходнаяНакладная".
- **ПараметрыОповещения** – как правило, это параметр события **ПоследЗаписи** формы объекта. Состав свойств структуры может быть расширен исходя из потребностей прикладной логики.
- **ОбъектСсылка** – ссылка на записываемый объект.

Пример:

```
&НаКлиенте
Процедура ПоследЗаписи(ПараметрыЗаписи)
    Оповестить("Запись_РасходнаяНакладная", ПараметрыЗаписи, Объект.Ссылка);
КонецПроцедуры
```

Затем в обработчике события **Оповещение** в форме динамического списка разместить код по обновлению динамического списка вида:

```
&НаКлиенте
Процедура ОбработкаПовещения(ИмяСобытия, Параметр, Источник)
```

```
Если Врг(ИмяСобытия) = Врг("Запись_РасходнаяНакладная")
    Или Врг(ИмяСобытия) = Врг("Запись_НакладнаяНаПеремещение") Тогда
        Элементы.СписокНакладных.Обновить();
КонецЕсли;
```

КонецПроцедуры

При этом имена событий следует проверять явным образом, чтобы исключить неоправданное обновление списка.

3.3. При изменении сразу нескольких объектов в параметре **ОбъектСсылка** метода **Оповестить** следует передавать значение **Неопределено**.

Пример:

```
Оповестить("Запись_РасходнаяНакладная", ПараметрыЗаписи, Неопределено);
```

См. также

- [Особенности реализации команд для форм списков](#)

Реквизит Ссылка и признак "Использовать всегда" в динамических списках объектов

Область применения: управляемое приложение.

#std702

Рекомендация (полезный совет)

1. С помощью команды **Еще - Изменить форму** пользователь может добавить отображение любых реквизитов объектов в динамическом списке. Для того чтобы пользователь смог воспользоваться этой возможностью, в динамических списках объектов ссылочных типов (справочников, документов и т.п.) рекомендуется добавлять в список поле **Ссылка**, у которого отключать пользовательскую видимость.

Кроме того, при использовании в объектах различных механизмов платформы 1С:Предприятие и **Библиотеки стандартных подсистем** наличие реквизита **Ссылка** в динамических списках может быть критичным для работоспособности этих механизмов. Например:

- при использовании механизма характеристик платформы 1С:Предприятие (или подсистемы "Свойства" **Библиотеки стандартных подсистем**) в динамические списки рекомендуется всегда выводить реквизит **Ссылка**, т.к. характеристики (дополнительные реквизиты и сведения) можно вывести в колонки списка только используя настройку в пользовательском режиме;
- при разработке команд печати (подсистема "Печать" **Библиотеки стандартных подсистем**) реквизит **Ссылка** с признаком "Использовать всегда" становится обязательным для корректной работы этих команд.

2.1. При обращении к данным строкам динамического списка в коде (например, в обработчике **ПриАктивацииСтроки**, **ПриПолученииДанныхНаСервере** и др.) необходимо учитывать, что в целях оптимизации данные невидимых колонок не получаются. При этом видимостью может управлять, в том числе, пользователь.

2.2. Для колонок, у которых в свойствах таблицы формы, отображающей динамический список установлен признак "Использовать всегда" данные получаются всегда, вне зависимости от видимости.

Не рекомендуется устанавливать признак "Использовать всегда" для колонки реквизита динамического списка, кроме тех случаев, когда значение колонки обязательно для выполнения алгоритма. Например, колонка **Ссылка** всегда требуется для команды "Печать", но эта колонка может быть скрыта пользователем.

Запросы в динамических списках

Область применения: управляемое приложение, мобильное приложение.

#std732

При проектировании динамических списков в формах следует учитывать, что динамические списки предъявляют более высокие требования к скорости выполнения запросов, чем в других случаях (например, в отчетах, в процедурах обработки данных и пр.). Данные в динамических списках непосредственно отображаются пользователю, поэтому скорость вывода и обновления данных является критичной. В данной статье перечислены рекомендации, дополняющие общие сведения по оптимизации запросов (см. группу статей «Оптимизация запросов»).

1. Нужно стараться делать простые запросы для динамических списков. Для этого в первую очередь нужно оптимизировать архитектуру хранения данных, чтобы их было просто отображать в динамических списках.

Пример

В динамическом списке документов-распоряжений на отгрузку нужно вывести состояние отгрузки. Состояние зависит от остатков регистра накопления, и статусов двух документов другого типа.

НЕПРАВИЛЬНО

Пытаться разработать запрос для динамического списка, который будет учитывать всю сложную логику расчета состояния отгрузки.

ПРАВИЛЬНО

Сделать регистр сведений «Состояния отгрузки», в котором хранить уже рассчитанное состояние отгрузки. При этом расчет можно делать или в процессе проведения документов, которые могут влиять на состояние отгрузки, или отдельным регламентным заданием.

2. Необходимо выбрать один из трех режимов работы динамического списка:

- **Динамическое считывание данных включено (рекомендуется).** Используются запросы, выбирающие записи в количестве приблизительно соответствующем количеству видимых строк в таблице;
- **Динамическое считывание данных выключено, задана не виртуальная основная таблица или одна из следующих таблиц: СрезПервых, СрезПоследних, ЗадачиПоИсполнителю, КритерийОтбора, ДвижениеСубкonto.** Используются запросы, выбирающие по 1000 записей в буфер на сервере, по мере необходимости данные передаются на клиент. Менее эффективно, чем динамическое считывание;
- **Динамическое считывание данных выключено, основная таблица не задана.** Запрос выполняется «как есть». В буфере накапливаются данные, начиная с 1000 записей. Чем ближе к концу списка, тем больше записей. Можно использовать только для заведомо маленьких выборок.

3. При разработке динамического списка следует учитывать что запрос, который фактически будет сформирован к СУБД, зависит от предопределенных настроек отборов, порядка и группировки СКД. В частности это означает, что необходимо рассмотреть индексирование полей:

- По которым выполняется соединение в запросе;
- На которые наложены условия в запросе;
- Выбранных в качестве быстрых отборов;
- По которым выполняется упорядочивание или предусмотрена группировка;
- По которым ожидается, что пользователь будет часто упорядочивать (группировать).

При этом не следует индексировать все поля подряд «на всякий случай», так как избыточные индексы создают неоправданную нагрузку при записи данных.

См. также: [Несоответствие индексов и условий запроса](#), [Ограничения при использовании динамических списков](#)

4. Настоятельно не рекомендуется использовать конструкции, «утягивающие» запрос:

- конструкции РАЗЛИЧНЫЕ и СГРУППИРОВАТЬ ПО;
- конструкции ВЫБОР в предложении ГДЕ или в условиях соединения;
- упорядочивание по полу, полученному при помощи конструкции ВЫБОР, в том числе и пользовательское.

См. также: [Общие требования по разработке оптимальных запросов](#)

5.1 Соединяться в запросе следует только с небольшим количеством реальных таблиц (в оптимальном варианте в динамическом списке – только одна таблица, и она назначена основной).

Не рекомендуется выполнять соединения:

- с большим количеством реальных таблиц. Ориентироваться стоит на количество не более 4 таблиц;
- с вложенными запросами;
- с виртуальными таблицами.

См. также: [Запросы, выполняющие соединение с вложенными запросами или виртуальными таблицами](#)

5.2. Соединение с виртуальными таблицами допустимо в отдельных случаях, если запрос и архитектура данных удовлетворяют ряду условий:

- допустимо соединение с виртуальными таблицами СрезПоследних (СрезПервых), если регистр сведений содержит заведомо небольшое количество записей. Например, получение текущего курса валют по данным регистра сведений КурсыВалют;
- при обращении к виртуальной таблице будут использованы хранимые итоги регистра сведений (см. [Разрешение итогов для периодических регистров сведений](#));
- запрос к виртуальной таблице Остатки будет преобразован платформой в простое чтение хранимой таблицы итогов без группировок (см. [Эффективное обращение к виртуальной таблице «Остатки»](#)).

использование временных таблиц.

5.3.1. Временные таблицы в динамических списках рекомендуется использовать только тогда, когда они содержат заведомо небольшое количество записей. Иначе их использование неэффективно, т.к. значения временных таблиц в динамическом списке НЕ кешируются, а формируются при каждом считывании данных для заполнения списка.

5.3.2. В частности, если не удается переделать запрос динамического списка, используя виртуальные таблицы с ограничениями (см. п. 5.2), или вообще отказавшись от их использования, то вместо соединения с ними следует использовать соединения с временными таблицами.

5.3.3. Если последний запрос динамического списка выбирает данные только из ранее созданной временной таблицы, то это уже не динамический список и следует перепроектировать его запрос и, скорее всего, метаданные, используемые запросом.

Разыменование ссылочных полей составного типа в языке запросов

5.5. При работе со списком под неполными правами, в которых настроены ограничения доступа к данным (RLS) к таблицам, участвующим в запросе*, также автоматически присоединяются условия ограничения доступа к данным, которые замедляют работу списка. Именно в этом режиме работы следует проверять скорость работы динамических списков.

* примечание: к тем таблицам, к которым предусмотрен RLS в конфигурации.

6. Предусмотреть оптимизацию при работе с полями составного типа

- При использовании в соединениях, отборах, упорядочивании и других конструкциях составного поля, необходимо чтобы состав типов данного поля определялся только ссылочными типами. Подробнее см.: [Ограничения на использование реквизитов составного типа](#);
- Если известно заранее, какого типа должно быть получено поле составного типа, то его необходимо выражать.

Например:

ВЫРАЗИТЬ(ДанныеДокумента.ДокументОснование КАК Документ.АктВыполненныхРабот)

См. также: [Разыменование ссылочных полей составного типа в языке запросов](#)

7. Запрос динамического списка рекомендуется менять «на лету» на более оптимальный, если это возможно.

Например, если изменение настройки позволяет переписать запрос динамического списка так, что он будет обращаться к другим метаданным, что позволит выполнятся ему быстрее.

См. также: [Программное переопределение текстов запросов динамических списков](#)

8. Если применение вышеизложенных рекомендаций не возможно, либо оно не дает должного эффекта, то можно рассмотреть следующие пути оптимизации:

8.1. Отказаться от части возможностей динамического списка.

Например:

- От вывода не столь значимой информации, получение которой приводит к усложнению запроса;
- От реализованных сервисных возможностей по группировке списка.

8.2. Осуществлять вывод данных не в динамический список, а в таблицу или дерево значений.

При этом появятся возможности оптимизации недоступные для динамических списков, такие как использование привилегированного режима и т.п..

Данный способ применим, если выполняется одно из условий:

- Исходных данных заведомо мало (десятка-сотни записей).
- Обязательные отборы, накладываемые на список, гарантируют, что данных в один момент времени записей выводится мало.
- Порционность вывода данных организована другими средствами (вручную), например, как в результатах полнотекстового поиска.

9. В случаях, когда в динамическом списке требуется отображение вспомогательных колонок, по которым не требуется отбирать (в том числе через механизмы поиска), сортировать и группировать, и затруднительно, неэффективно или невозможно выполнить получение данных с помощью основного запроса, рекомендуется воспользоваться обработчиком **ПриПолученииДанныхНаСервере** таблицы управляемой формы. Например, колонки **Курс на сегодня**, **Кратность** в списке валют и т.п.

Для эффективной работы обработчика **ПриПолученииДанныхНаСервере** следует выбирать всю вспомогательную информацию одним запросом сразу для всех отображаемых строк, которые передаются в этот обработчик после выполнения основного запроса динамического списка.

Кроме того, для корректной работы динамического списка требуется явно ограничить выполнение отбора, сортировки и группировки по вспомогательным колонкам с помощью методов динамического списка **УстановитьОграниченияИспользованияВГруппировке**, **УстановитьОграниченияИспользованияВПорядке** и **УстановитьОграниченияИспользованияВОтборе**.

Поле "Дата" в списках

Область применения: управляемое приложение, мобильное приложение.

#std745

Сокращенное представление поля **Дата** в списках, содержащих реквизит с составом даты **Дата и время**.

В целях оптимизации места на форме, рекомендуется изменить представление поля **Дата** в списках:

1. Установить ширину поля **Дата без времени: 9 пунктов**
2. Отображать дату документов в виде даты без времени: **01.01.2015**
3. Для систем оперативного учета, рекомендуется дату сегодняшних документов отображать в виде времени: **09:46**

Например:

Рекомендуется			Не рекомендуется		
Номер	Дата	Сумма	Номер	Дата	Статус
ДС00-000004	10.17	36 420	УУ00-000001	29.04.2015 14:42:53	Реализовано
ТД00-000027	26.06.2015	39 159	ТД00-000032	30.04.2015 11:12:30	Реализовано
ДС00-000003	26.06.2015		ТД00-000035	30.04.2015 16:19:33	Реализовано
ДС00-000002	25.06.2015		ТД00-000048	01.05.2015 23:59:59	Реализовано
ДС00-000001	24.06.2015		ТД00-000036	05.05.2015 12:00:00	Реализовано
ТД00-000026	15.05.2015	11 692	ТД00-000042	10.05.2015 12:00:03	Реализовано
ТД00-000025	13.05.2015	336,1	ТД00-000045	11.05.2015 12:00:00	Реализовано
ТД00-000022	12.05.2015	39 159	ТД00-000038	12.05.2015 12:00:00	Реализовано
ТД00-000024	10.05.2015	14 260	ТД00-000037	12.05.2015 14:01:08	Реализовано

Пример кода:

```
&наСервере
Процедура ПриСозданиинаСервере(Отказ, СтандартнаяОбработка)
    СтандартныеПодсистемыСервер.УстановитьУсловноеОформлениеПоляДата(ЭтотОбъект, "Список.Дата", Элементы.Дата.Имя);
КонецПроцедуры
```

Для автоматического изменения свойств полей выбора можно воспользоваться приложенной [обработкой](#).

См. также

- [Даты: требования по локализации](#)

Программное переопределение текстов запросов динамических списков

1. Если текст запроса динамического списка переопределяется в коде, то рекомендуется придерживаться правил, описанных ниже. Это нужно для того, чтобы:

- разработчики, глядя на запрос в динамическом списке, могли однозначно понять, что он переопределяется в коде конфигурации;
- не снижать производительность при открытии формы, установке текста запроса, основной таблицы и признака динамического считывания данных.

1.1. В редакторе запроса динамического списка следует устанавливать полноценный наиболее часто выполняемый текст запроса (запрос по умолчанию), чтобы при открытии формы, в большинстве случаев, не происходила программная установка текста запроса, снижающая производительность. Исключением является случай, когда текст запроса собирается программно.

ПРАВИЛЬНО:

Полноценный запрос:

ВЫБРАТЬ

НоменклатураПереопределяемый.Ссылка КАК Ссылка

ИЗ

Справочник.Номенклатура КАК НоменклатураПереопределляемый

НЕПРАВИЛЬНО:

Запрос, требующий обязательной замены в коде конфигурации, т.к. не содержит секции ИЗ:

ВЫБРАТЬ

ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка) КАК Ссылка

1.2. Псевдонимы таблиц должны заканчиваться постфиксом «Переопределляемый», чтобы визуально было заметно, что этот запрос может переопределяться в коде конфигурации.

ПРАВИЛЬНО:

ВЫБРАТЬ

НоменклатураПереопределляемый.Ссылка КАК Ссылка

ИЗ

Справочник.Номенклатура КАК НоменклатураПереопределляемый

НЕПРАВИЛЬНО:

ВЫБРАТЬ

Номенклатура.Ссылка КАК Ссылка

ИЗ

Справочник.Номенклатура КАК Номенклатура

1.3. Установка текста запроса и основной таблицы при первичной инициализации динамического списка (в При Создании На Сервере) должна выполняться до любого обращения к настройкам этого списка (в т.ч. параметрам), чтобы не снижалась производительность. В остальных случаях, если требуется изменить текст запроса и основную таблицу, между этими действиями не следует обращаться к настройкам (можно до них или после). При наличии БСП, следует использовать процедуру Общего Назначения.Установить Свойства Динамического Списка(). Это необходимо для повышения производительности и возможности автоматического сбора переопределяемых текстов запросов динамических списков.

ПРАВИЛЬНО:

```
СвойстваСписка = ОбщегоНазначения.СтруктураСвойствДинамическогоСписка();  
СвойстваСписка.ОсновнаяТаблица = "Справочник.Номенклатура";  
СвойстваСписка.ДинамическоеСчитываниеДанных = Истина;  
СвойстваСписка.ТекстЗапроса = ТекстЗапроса;  
ОбщегоНазначения.УстановитьСвойстваДинамическогоСписка(Элементы.Список,  
СвойстваСписка);  
Список.Параметры.УстановитьЗначениеПараметра("Параметр1", 42);
```

НЕПРАВИЛЬНО:

```
Список.ТекстЗапроса = ТекстЗапроса;  
Список.Параметры.УстановитьЗначениеПараметра("Параметр1", 42);  
Список.ОсновнаяТаблица = ОсновнаяТаблица;
```

В этом случае снижается производительность из-за того, что при изменении текста запроса или основной таблицы сбрасывается источник доступных настроек и при обращении к Список.Параметры источник инициализируется заново.

2. Переопределяемые тексты запросов динамического списка можно размещать в любых модулях конфигурации, но следует следить за тем, чтобы текст запроса в динамическом списке был точно таким же, как и переопределяемый текст запроса по умолчанию (наиболее часто используемый).

Этого можно добиться, например, если:

- изменить текст запроса по умолчанию в коде конфигурации;
- после изменения, скопировать текст запроса в динамический список;
- или наоборот (изменить в динамическом списке, скопировать в код конфигурации).

См. также

- [Запросы в динамических списках](#)

Информирование пользователя

Область применения: управляемое приложение.

#std400

1. Случай, когда необходимо довести информацию до пользователя:

- Множество сообщений (протокол) по поводу невозможности выполнения того или иного действия.
- Множество сообщений (протокол) по поводу выполнения того или иного действия.
- Информация об ошибке выполнения действия.
- Информация об успешном выполнении того или иного действия при работе в форме.
- Информация об успешном выполнении того или иного действия в условиях отсутствия формы.

1.1. Информация об ошибках, выявленных при проверке заполнения, должна выводиться в панели сообщений формы. Например, такой способ информирования пользователя предлагает платформа для наименования справочников. Проверки заполнения можно разделить на два случая:

А) Проверку заполнения реквизитов объекта следует выполнять вне транзакции записи объекта.

Пример:

- При попытке записать объект необходимо выдать ряд сообщений о его неправильном заполнении;
- При выполнении команды "Пересчитать цены" необходимо выдать сообщения об ошибках пересчета цен товаров, выбранных в документе.

Средства для реализации:

- Свойство Проверка заполнения
- [Обработка Проверки Заполнения](#) (в этом случае для формирования сообщений об ошибках следует применять объект Сообщение Пользователю)

Б) Проверку целостности объекта и связанных с ним данных следует выполнять в транзакции записи объекта.

Пример:

- Проверка складских остатков товаров и выдача нескольких сообщений о нехватке товаров

Средства для реализации:

- [При Записи](#)
- Для формирования сообщений об ошибках следует применять объект Сообщение Пользователю
- Для отказа от записи объекта устанавливать параметр Отказ = Истина

При этом

- если ошибочная ситуация носит исключительный характер и делает бессмысленными все остальные проверки, которые выводят другие блокирующие сообщения, то следует использовать **ВызватьИсключение** (см. ниже п.1.3, пример №2). Но если пользователь может оперативно устраниить эту ошибочную ситуацию (то есть это ошибка ввода данных пользователем, а не ошибка настройки системы), то следует устанавливать параметр **Отказ** в значение **Истина** и выводить сообщения пользователю.
- следует минимизировать число проверок, которые выполняются во время записи объекта. Во время записи должны выполняться только проверки, которые носят транзакционный характер. Например, если подразделение сотрудника можно сменить в любой момент времени и при этом не проверяется, существуют ли уже документы, в которых выбран данный сотрудник, то также не имеет смысла проверять при записи документа соответствие выбранного сотрудника и заданного подразделения. Такую проверку следует выполнять в процедуре **ОбработкаПроверкиЗаполнения** до начала транзакции записи.

1.2. Информация о протоколе выполненных действий

может выводиться в отдельном поле в форме.

Пример:

- При выполнении команды формы "Прокомментировать расчет" выдать информацию об использованных при расчете данных и промежуточных результатах расчета

Средства для реализации:

- Формирование сообщений в отдельном реквизите формы и изменение видимости поля в форме;
- Открытие отдельной формы со списком сообщений.

1.3. Информация об ошибке

должна доводиться до пользователя в отдельном диалоге. Например, такой способ информирования пользователя предлагает платформа при выводе системных сообщений об ошибке: сообщения о неуникальности кода, неуникальности записи регистра сведений, незаданной дате документа и т.п.

Пример №1:

- При выполнении локальной команды формы "Заполнить скидку" необходимо сообщить, что еще не выбран контрагент
- При выполнении локальной команды "Рассчитать зарплату" необходимо сообщить, что в системе еще не задан курс валюты на текущий месяц.

Способ реализации:

- Проверять необходимые значения в коде формы и использовать метод глобального контекста **ПоказатьПредупреждение**.
- В случае если есть вариант решения проблемы, рекомендуется использовать функцию **ПоказатьВопрос**. Например:

```
Оповещение = Новый ОписаниеОповещения("ПослеЗакрытияВопроса", ЭтотОбъект);
ПоказатьВопрос(Оповещение, НСтр("гу='В этом месяце еще не задан курс валюты. Использовать курс за прошлый месяц?'"), ...))
```

Пример №2:

- При выполнении функции расчета кросс-курса валюты по отношению к управляемой валюте сообщить, что не задан курс валюты, не задана управляемая валюта и т.п.;
- Сообщить о запрете при записи документа в закрытом периоде;
- Выдать сообщение об ошибке и предотвратить запись объекта, для которого загружены некорректные правила регистрации на узлах плана обмена.

Способ реализации:

- Вызвать исключение. Например:

Правильно:

```
ВызватьИсключение НСтр("гу='Не установлена управляемая валюта в настройках программы.'");
```

Правильно:

```
Если Не ЗарегистрироватьИзмененияНаУзлахПлановОбмена() Тогда
    ВызватьИсключение НСтр("гу = 'Не удалось зарегистрировать изменения на узлах планов обмена из-за некорректных правил регистрации.
    Обратитесь к администратору.'");
КонецЕсли;
```

1.4. Информация об успешном выполнении действия в форме

должна выводиться в случае, если факт выполнения команды не очевиден для пользователя. Система должна выдавать ту или иную реакцию на любую команду. Например, неправильно "проглатывать" молча нажатие на кнопку формы.

Доводить информацию до пользователя рекомендуется как при помощи модального диалога (например, процедуры **Предупреждение**), так и при помощи метода **ПоказатьОповещениеПользователя**. Применять процедуру **Предупреждение** следует в том случае, если требуется "приостановить" работу пользователя и обратить его внимание на результат выполнения команды.

Пример:

- При выполнении команды формы "Проверить заполнение" необходимо сообщить, что проверка ошибок не обнаружила;
- При выполнении команды формы "Ограничить остатками на складе" необходимо сообщить, что количество товаров проверено и не было уменьшено;
- При выполнении команды формы "Принять изменения" необходимо сообщить, что обработаны все пять выбранных документов.

Способ реализации:

- Использовать процедуру **Предупреждение**

Вместе с тем, не следует информировать пользователя об очевидно выполненном действии. Например, не следует сообщать об успешном выполнении команды "Заполнить" в условиях, когда заполняемый список товаров у пользователя "на виду", и пользователь и так отчетливо видит результат выполнения команды.

1.5. Информация об успешном выполнении действия в условиях отсутствия формы

на экране должна также выводиться в том случае, когда для пользователя может оказаться неочевидным тот факт, что действие выполнено.

1.5.1. Использовать процедуру **ПоказатьОповещениеПользователя** для уведомления о завершении действия.

Например, при выполнении стартовой обработки информационной базы сообщить, что обработка выполнена успешно:

```
ПоказатьОповещениеПользователя(
    НСтр("гу = 'Обновление на версию 2.0 успешно выполнено.'"),
    НСтр("гу = 'Переход на новую версию'"),
    БиблиотекаКартинок.Информация32);
```

1.5.2. Использовать процедуру **Состояние** при изменении состояния выполняемого действия на клиенте и после завершения действия (без последующего вызова **ПоказатьОповещениеПользователя**).

Например, при обработке нескольких файлов на клиенте сообщать: «Обработан файл 1 из 5...», «Обработан файл 4 из 5...», «Обработка файлов завершена»:

```
Состояние(СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку (
    НСтр("гу = 'Обработан файл %1 из %2.'"), СчетчикФайлов, КоличествоФайлов),
    СчетчикФайлов/КоличествоФайлов*100);

...;
```

```
Состояние(НСтр("гу = 'Обработка файлов успешно завершена.'"));
```

В то же время для серверных операций не рекомендуется выполнять оповещение с помощью процедуры **Состояние**, т.к. в этом случае потребуются многократные вызовы сервера для передачи прогресса. См. [Длительные операции на сервере](#).

См. также

- [Сообщения пользователю](#)
- [Перехват исключений в коде](#)
- [Работа с параметром «Отказ» в обработчиках событий](#)

Ограничение на использование метода Сообщить

Область применения: управляемое приложение.

#std418

Для вывода сообщений пользователю во всех случаях следует использовать объект **СообщениеПользователю**, даже когда сообщение не «привязывается» к некоторому элементу управления формы. Метод **Сообщить** применять не следует.

Методическая рекомендация (полезный совет)

При использовании в конфигурации **Библиотеки стандартных подсистем** рекомендуется использовать процедуру **СообщитьПользователю** общего модуля **ОбщегоНазначения** или **ОбщегоНазначенияКлиент**, которая работает с объектом **СообщениеПользователю**.

См. также

- [Информирование пользователя](#)
- [Сообщения пользователю](#)

Установка внешних компонент и расширений платформы

Область применения: управляемое приложение.

#std700

1.1. Установка внешних компонент и расширений платформы должна быть интерактивной. Пользователь должен самостоятельно принять решение об установке. В диалоге установки должно быть указано, для чего нужна компонента (расширение) и что не будет работать, если ее не устанавливать.

Например, неправильно использовать конструкции вида

Если Не ПодключитьВнешнююКомпоненту(...) Тогда
УстановитьВнешнююКомпоненту(...)

Правильно задавать пользователю вопрос в явном виде:

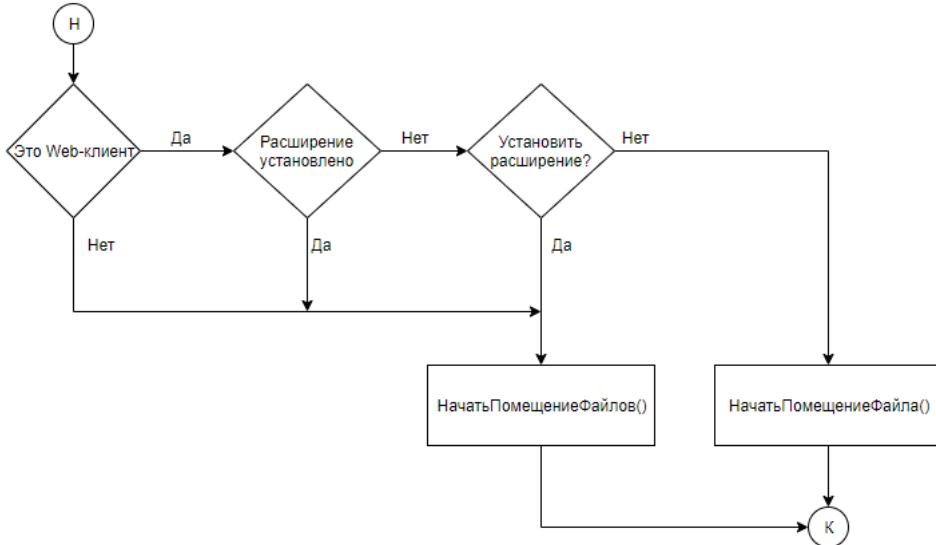
Для продолжения работы требуется установить внешнюю компоненту, которая позволит работать с отчетностью. Для установки компоненты нажмите "Установить". После завершения установки нажмите "Продолжить".

1.2. Рекомендуется выводить предложение об установки компоненты (расширения) перед выполнением прикладного действия.
Например:

- Пользователь воспользовался командой «Отправить отчет»
- Для этого конфигурации необходимо, чтобы была установлена какая-либо внешняя компонента.
- Конфигурация проверяет, установлена ли компонента.
- Если компонента не установлена, отображает пользователю информацию о том, что для отправки отчета нужно установить компоненту и кнопку, вызывающую установку компоненты.
- Пользователь нажимает на кнопку, выполняется установка.
- После установки пользователь нажимает на кнопку «Продолжить отправку отчета»
- Программа продолжает отправлять отчет.

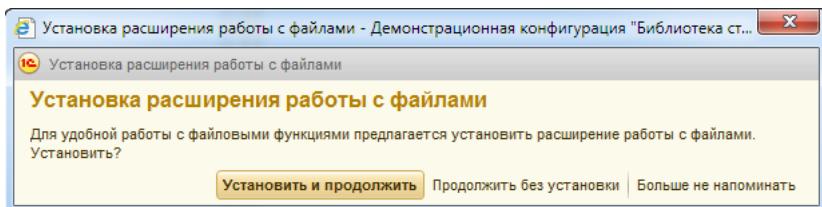
Такой сценарий позволяет обеспечить, чтобы компоненты (расширения) устанавливались без проблем на всех поддерживаемых браузерах, в том числе, в браузере **FireFox**.

Другой пример. Предложение об установке расширения работы с файлами при загрузке файла из файловой системы:



1.3. При использовании в конфигурации **Библиотеки стандартных подсистем** для вывода предложения об установке расширения работы с файлами следует использовать следующие процедуры общего модуля **ФайловаяСистемаКлиент** в следующих сценариях:

- ВыбратьКаталог** вместо метода **Показать** объекта **ДиалогВыбораФайла** с заданным режимом работы **ВыборКаталога**;
- ЗагрузитьФайл** вместо методов глобального контекста **ПоместитьФайл**, **НачатьПомещениеФайла**, а также вместо метода **Показать** объекта **ДиалогВыбораФайла** с заданным режимом работы **Открытие**;
- ЗагрузитьФайлы** вместо методов глобального контекста **ПоместитьФайлы**, **НачатьПомещениеФайлов**, а также вместо метода **Показать** объекта **ДиалогВыбораФайла** с заданным режимом работы **Открытие**;
- ОткрытьФайл** вместо метода глобального контекста **ЗапуститьПриложение** для открытия файла, ассоциированного с некоторым приложением;
- СохранитьФайл** вместо метода глобального контекста **ПолучитьФайл** или метода **Показать** объекта **ДиалогВыбораФайла** с заданным режимом работы **Сохранение**;
- СохранитьФайлы** вместо методов глобального контекста **ПолучитьФайлы**, **НачатьПолучениеФайлов**, а также вместо метода **Показать** объекта **ДиалогВыбораФайла** с заданным режимом работы **Сохранение**.
- В остальных случаях, для вывода предложения об установке расширения работы с файлами следует использовать процедуру **ПодключитьРасширениеДляРаботыСФайлами**.



2. В прикладном решении должны быть предоставлены инструменты для установки пользователем внешних компонент и расширений в любой момент работы. Таким образом, их можно установить не только в ходе решения какой-то задачи, но и в виде отдельного действия (из некоторого административного режима).

Методическая рекомендация (полезный совет)

При использовании в конфигурации **Библиотеки стандартных подсистем** для установки расширения для работы с файлами предназначена общая команда **УстановитьРасширениеРаботыСФайлами**, которую рекомендуется размещать в форме персональных настроек пользователя (см. общую форму **_ДемоМоиНастройки** в демонстрационной конфигурации). В этой же форме рекомендуется размещать команды по установке внешних компонент, которые могут потребоваться пользователю при его работе.

См. также

- [Общие требования к конфигурации](#)

Общие рекомендации

Область применения: управляемое приложение.

#std566

См. также: [Размеры экрана \(8.3\)](#)

1. Типовым разрешением экрана считается 1280x768 pt. Все размеры в стандартах приведены с учетом этих значений. Условия эксплуатации:

- Основное окно 1С:Предприятия растянуто на весь экран
- Видна панель задач операционной системы

2. Разработку (конфигурирование) нужно вести в стандартном разрешении - 96 DPI.

3. При разработке управляемого интерфейса не рекомендуется вносить массовые изменения, нарушающие умолчание платформы, если иное не указано в этом руководстве по стилю или других стандартах.

Например, для контекстных меню лучше использовать умолчание платформы, но в отдельных специфических случаях допускается внесение изменений.

Рекомендация (полезный совет)

4. При проектировании интерфейсов рекомендуется использовать [Чек-лист проверки интерфейса](#). Проверка формы по чек-листу позволяет быстро выявить грубые ошибки в интерфейсе и выяснить, обеспечивается ли приемлемый уровень качества.

Командный интерфейс

Область применения: управляемое приложение.

#std600

См. также: [Общие принципы построения командного интерфейса \(8.3\)](#)

Командный интерфейс (КИ) – средство навигации пользователей по функциональности конфигурации. КИ создается панелью разделов, панелью навигации и панелью действий. Каждая из панелей имеет свое назначение, но все вместе они создают пространство команд – возможностей для пользователя.

При проектировании КИ следует учитывать, что все три навигационных элемента связаны контекстом и предназначены:

- для повышения эффективности выполнения повседневной работы;
- для быстрого освоения программы.

Качество конфигурации во многом зависит:

- от того насколько удачно это пространство будет организовано;
- насколько оно будет соответствовать представлениям пользователя о назначении этой конфигурации.

Критериями хорошо проектированного КИ являются:

- скорость работы;
- скорость обучения;
- субъективная удовлетворенность пользователя.

Панель разделов

См. также: [Панель разделов \(8.3\)](#)

Панель разделов (ПР) – оглавление конфигурации.

Разделы должны соответствовать реальным областям деятельности или участкам работ так, как их понимают пользователи.

Состав разделов рекомендуется проектировать так, чтобы в типовых комбинациях ролей не появлялась полоса прокрутки. Для этого емкость панели должна быть не более 8-10 разделов.

Что следует учитывать при проектировании:

- Панель разделов является «лицом» программы и имеет большое значение на этапе освоения, поэтому состав и порядок разделов следует проектировать с особой тщательностью.
- Предполагается, что при выполнении задач связанных с определенной деятельностью пользователь будет проводить большую часть времени в каком-то одном разделе. Т.е. раздел является устойчивым режимом работы. Задачи, относящиеся к одной деятельности или к одному участку работ, должны решаться в рамках одного раздела – переключения между разделами должны быть минимизированы.
- Интерфейс раздела нужно проектировать так, чтобы он содержал в себе все необходимые для работы команды и функции. При этом частотные или важные команды следует «вытаскивать наверх» – в начало панели навигации и панели действий.
- При проектировании панели учитывайте и используйте доступность разделов (подсистем) по ролям.
- Для интерфейса администратора допустимо наличие полосы прокрутки в панели.
- По умолчанию разделы располагаются в алфавитном порядке. Рекомендуется определять порядок в зависимости от приоритета каждого раздела: от наиболее частотного и значимого до второстепенного и не часто используемого.
- Последним всегда должен быть раздел для администрирования, настройки и выполнения сервисных действий.
- Не рекомендуется делать раздел с названием «Сервис».

Т.к. он будет перекликаться с меню «Сервис» и группой «Сервис» в ПД. Альтернативой могут быть такие названия как «Сервисные возможности», «Дополнительные возможности», «Прочее», «Сервисные функции», «Служебные функции» и т.д.

Названия разделов

См. также: [Панель разделов \(8.3\)](#)

- Общая длина названия не должна превышать примерно 50 знаков с учетом пробелов (примерно 150 точек при 96 DPI).
- При выборе названия (синонима) нужно учитывать, что панель разделов отображает максимум две строки названия с автоматическим переносом строк, но без переноса слов, т.е. слова не разрываются.
- Для того чтобы название раздела смотрелось симпатично, рекомендуется использовать такие комбинации слов:
 - одно-два средних слова;
«Финансы», «Зарплата и персонал»
 - одно среднее и одно короткое;
«Учет времени»
 - одно длинное и одно короткое;
«Сервис и администрирование»
 - два коротких и одно длинное.
«Работы, услуги, производство»
- **По возможности не используйте длинные слова.**
Например, «гидролесомелиорация» или «делопроизводство».
- Выбирайте названия примерно одного размера по ширине так, чтобы они смотрелись единообразно и ровно.
- Давайте разделам конкретные (не двусмысленные) запоминающиеся названия.
- Используйте в названиях только общепринятые и соответствующие целевой аудитории сокращения и аббревиатуры, например, «НДС» или «МСФО».

Но если это возможно, лучше обходиться без сокращений.

См. также:

[Оформление текстов](#)

Картинки разделов

См. также: [Панель разделов \(8.3\)](#)

- Для каждого раздела в ПР назначьте картинку размером 48x48 pt. Формат PNG с переменным альфаканалом.
- Картинки следует делать разными по начертанию и по ведущим цветам. Это нужно для лучшей запоминаемости. Но при этом они должны быть нарисованы в одном стиле, с одинаковым углом поворота, выравниванием, направлением света и т.д.
- Следует проверять, чтобы в режиме отображения «только картинки» ПР выглядела ровно и гармонично.

См. также:

Подсказки для интерфейсных подсистем

См. также: [Панель разделов \(8.3\)](#)

- Подсказки для ПР не являются обязательными – назначение раздела должно быть однозначно понято из его названия.
Название раздела должно быть настолько интуитивно понятно, чтобы пользователь, в поисках нужной ему информации, сразу понял, в каком разделе ее следует искать.
- Подсказка может дополнительно расшифровывать содержимое раздела, например, с помощью краткого перечисления входящих в него функций.
- Справка для разделов ПР не является обязательной.

См. также:

Панель навигации основного окна

Навигация внутри раздела (8.3)

Панель навигации (ПН) представляет собой оглавление раздела.

Основная ценность панели заключается в том, что она всегда видна и позволяет быстро перейти к нужному списку или к рабочему месту.

Размеры панели по умолчанию – 243x589 pt.

Количество обычных команд – 31 (без группировки, выделения важных, без полосы прокрутки).

Названия команд

Область применения: управляемое приложение.

#std573

См. также: [Навигация внутри раздела \(8.3\)](#)

- По возможности, названия команд должны помещаться в ПН с учетом ее стандартной ширины ПН (234 pt).
- Важные команды – примерно 30 символов (т.к. они отображаются жирным шрифтом).
- Обычные команды – примерно 38 символов.
- Лучше, если названия команд начинаются с разных слов, так их удобнее искать с взглядом.

Следует учитывать, что ширина ПН может быть уменьшена пользователем поэтому команды должны отличаться друг от друга уже в первых символах, но в тоже время должны быть понятны.

Порядок команд

- По умолчанию команды располагаются в алфавитном порядке. Рекомендуется менять порядок так, чтобы частотные или первостепенные команды располагались сверху.

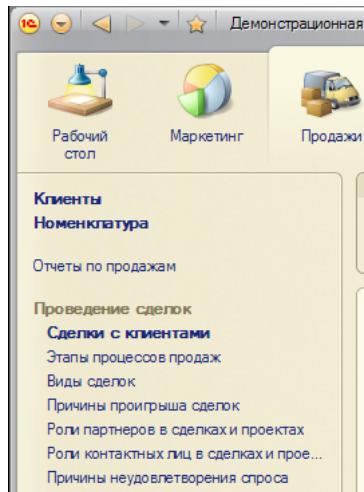
Группа команд «Важное»

Область применения: управляемое приложение.

#std623

См. также: [Навигация внутри раздела \(8.3\)](#)

«Важное» – команды переходов в ПН, выделенные жирным шрифтом. Могут выступать как отдельными элементами ПН, так и входить в группы.



- Наличие группы команд «Важное» не является обязательным.
- Использование необходимо в случаях прогнозируемой частотности переходов по той или иной команде. Это позволит пользователям быстрее ориентироваться в ПН.
- Выделяйте не более 5 важных команд.

При большем количестве команд ПН выглядит громоздко и выделение команд жирным шрифтом действует не так эффективно.

Группировка команд

Область применения: управляемое приложение.

#std574

См. также: [Навигация внутри раздела \(8.3\)](#)

- Если команд в группе «Обычное» получается много, группируйте их по смыслу. Для этого следует пересмотреть содержимое подсистемы и разбить его на подсистемы так, чтобы в группе было от 4 до 9 команд.
Критерием количества команд может служить оценка емкости кратковременной памяти человека – 7± 2.
- Состав команд следует проектировать так, чтобы по умолчанию в панели навигации не было полосы прокрутки.
Скрывайте нечастотные команды – пользователь, если потребуется, сам их себе включит.
- По возможности избегайте появления в панели навигации групп, состоящих всего из одной команды.
Помните, что при построении командного интерфейса учитываются роли и это может привести к тому, что для каких-то пользователей группа станет «вырожденной», т.е. состоящей всего из одной команды.
- Не рекомендуется делать глубину вложенности групп команд более 2.
При этом помните, что по умолчанию на втором уровне ширина текста команды будет меньше - 36 символов.
- При разработке учитывайте, что по умолчанию все группы раскрыты.
- Названия групп в ПН не должны пересекаться с системными группами, например, «Создать», «Отчеты», «Сервис», «См. также».

Группа «См. также»

См. также: [Навигация внутри раздела \(8.3\)](#)

- Наличие группы не является обязательным.
- В группу «См. также» не следует помещать команды по принципу «на всякий случай» или «вдруг пригодится».
- Группа «См.также» предназначена для того, чтобы обеспечить горизонтальные связи между подсистемами – в нее следует относить важные, полезные для пользователя команды, которые напрямую не относятся к текущему разделу, но могут быть востребованы в некоторых случаях.

Команды, размещаемые в панели навигации

См. также: [Навигация внутри раздела \(8.3\)](#)

- В ПН следует помещать то, без чего нельзя обойтись в контексте текущего раздела: команды для перехода к рабочим местам, спискам, специальным обработкам. Под специальными командами понимаются обработки похожие на обычные списки, например, «Журнал регистрации», но не «Удаление помеченных объектов».
- Обязательно размещайте команды перехода к спискам «первичных» данных.
Списки первичных документов/данных, имеющих самостоятельную ценность (заказы покупателей, расходные накладные). Первичные – такие документы, с которых начинаются бизнес-процессы.
Формы списков первичных данных при этом следует оптимизировать под соответствующие задачи пользователей, например поиск, или отбор необработанных заявок.
- Для документов рекомендуется:
 - помещать в панель навигации и команды журналов и команды списков;
 - команды перехода к спискам делать по умолчанию невидимыми.
- Если объект по логике второстепенен, подчинен другому объекту, то его можно вообще не выносить в командный интерфейс – например, подчиненный регистр сведений. Если же есть независимый регистр сведений, и он является важным для пользователя, то его следует поместить в панель навигации.
- НЕ размещайте в панели навигации команды открытия обычных отчетов.
Это связано с тем, что типовые отчеты должны открываться во вспомогательном окне со своим интерфейсом.
- Если отчет по сути является рабочим местом для выполнения тех или иных функций, то нужно сделать специальную навигационную команду открытия этого отчета и разместить ее в ПН.

Панель действий

См. также: [Навигация внутри раздела \(8.3\)](#)

Панель действий (ПД) – место, где пользователи всегда смогут найти самые востребованные команды в контексте текущего раздела.

Она также служит своеобразным введением, «рассказывает» о том, что в этом разделе можно сделать.

При разработке состава команд следует отталкиваться от задач пользователей, например, «продажа товаров», «работа с внутренними документами» и т.д. Спектр этих задач формирует список команд, среди которых можно выделить наиболее частотные, вероятные. Их и нужно разместить в ПД.

Цель ПД – обеспечить возможность быстро создавать новые объекты, выполнять типовые обработки или строить популярные отчеты, не выполняя переключения в тот или иной интерфейс в ПН.

Когда ПД не нужна:

- Если нельзя выделить частотные команды или их очень мало.
- Когда все частотные команды контекстные.
Например, команды создания групп не рекомендуется выносить в ПД, т.к. они почти всегда контекстны.
- Если выполнение действий рекомендовано разработчиком через специальные формы (рабочие места).
Например, перед созданием заказа поставщику нужно видеть общую картину заказов по организации.

Размеры ПД

При проектировании состава команд ПД, необходимо учитывать следующее:

- По умолчанию в видимой области ПД помещается не более 15-ти средних команд.
Под средней подразумевается команда, название которой состоит из 15-20 символов.
- Максимальная емкость ПД по высоте - 30 команд. Если команд больше, то появляется кнопка открытия полного списка команд.

Названия команд

- Текст стандартных команд (создание нового объекта, открытие отчета или обработки) формируется из синонима или представления соответствующего объекта метаданных. Это следует учитывать при присвоении имен и представлений объектам метаданных так, чтобы из текста команды было однозначно понятно ее назначение с учетом группы, в которую она входит (создать, отчет, сервис).
- Тексты пользовательских команд тоже должны соответствовать этой рекомендации.
- Не рекомендуется использовать очень короткие (менее 5 символов) и очень длинные (более 30) названия команд и синонимы (или представления) объектов метаданных.
- Лучше, если названия команд, синонимы или представления объектов метаданных начинаются с разных слов. Тогда соответствующие команды будут хорошо различаться в панели действий.

Группы команд в ПД

Если команду ПД нельзя отнести к одной из стандартных групп («Создать», «Отчеты», «Сервис»), то можно создавать свои группы. Рекомендуется минимизировать количество таких групп.

Подсказки в ПД

Подсказка и справка для команд в ПД не является обязательной, но рекомендуется.

Команды отчетов

- Если отчетов немного, то следует помещать команды для их открытия в панель действий. При этом нужно учитывать общее наполнение ПД, чтобы она не получилась громоздкой.
- Если отчетов много или если для управления ими пользователю нужен особый интерфейс, следует разработать специальное рабочее место по работе с отчетами.
Такое рабочее место может содержать, например:
 - Список отчетов и вариантов
 - Готовые настройки
 - Предварительный просмотр
- Команду вызова рабочего места рекомендуется поместить:
 - в панель навигации – если рабочее место должно открываться в главном окне;

- в группу «Сервис» панели действий - если рабочее место должно открываться в отдельном окне.

См. также: [Оформление текстов](#)

[Навигация внутри раздела \(8.3\)](#)

Содержание отчета

Область применения: управляемое приложение.

#std670

1. Показатели отчета

- Отчет может содержать одновременно несколько финансовых или нефинансовых, числовых или качественных показателей. Их состав должен соответствовать назначению отчета (цели анализа).
- В отчет включаются только те показатели, которые связаны общей целью анализа.
- В отчетах, реализованных без использования СКД, рекомендуется обеспечивать возможность расшифровки всех выводимых показателей.
- Отчеты не должны содержать данные, помеченные на удаление. Если такие данные требуется выводить в отчет, то их следует выделять явным образом (см. стандарт «Отчеты вида «таблица», «список» п.3.5).

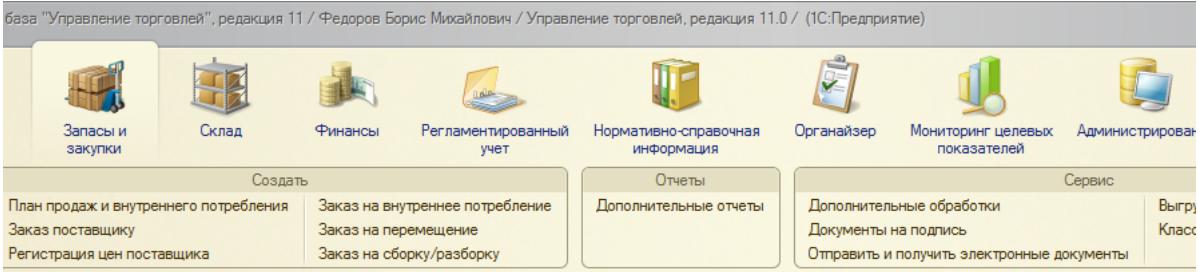
2. Валюты

- При включении показателя в отчет следует учитывать ситуации, когда он может принимать значения в разных валютах. Для сравнительного анализа валюта показателя должна быть единой в пределах отчета.
- Если назначение отчета предполагает анализ какого-либо показателя в различных валютах, то следует одновременно выводить значения показателя как в исходной валюте, так и в приведенной, единой валюте отчета. *Например, в отчете "Анализ расчетов с клиентами", расчеты с клиентами могут производиться в различных валютах. Для оценки объема задолженности по всем клиентам требуется приведение сумм задолженностей к единой валюте, например, к валюте управленческого или регламентированного учета (в зависимости от конфигурации).*

3. Контекстные и неконтекстные отчеты

- Отчеты и варианты отчетов могут быть предназначены для разных контекстов работы.

- Под неконтекстным отчетом подразумевается отчет, который запускается из командного интерфейса конфигурации (например, панели отчетов) вне какого-либо контекста для начала или продолжения анализа. Отчет запускается либо со стандартными, либо с сохраненными ранее пользовательскими настройками.



Отчеты по запасам и закупкам

Запасы

[ABC/XYZ - анализ номенклатуры по складам](#)

[Недостаточно товаров организаций](#)

[ABC/XYZ - анализ номенклатуры предприятия](#)

[Остатки товаров на складах](#)

[Анализ доступности товаров](#)

[Остатки товаров организаций](#)

Склад	Характеристика	В наличии	Резерв	Свободно
Бытовая техника	Вентилятор BINATONE ALPINE 160вт, напольный , Вентилятор JIPONIC (Тайв.), Вентилятор настольный Вентилятор настольный, Модель 901	194,000 6,000 6,000 10,000 10,000	37,000 6,000 6,000 4,000 10,000	157,000 6,000 6,000 6,000 10,000

- Под контекстным отчетом подразумевается отчет, который запускается из панели навигации какого-либо объекта учета (документа, элемента справочника), то есть при запуске происходит автоматическая настройка отборов/параметров отчета. Отчет в этом случае содержит информацию, имеющую отношение только к выбранному объекту учета.

Заказ клиента ТД-00...

Перейти
Карточка расчетов
Состояние выполнения
Состояние расчетов
Анализ цен
Движения документа
Задачи
Заказы клиентов с сайта
Мои заметки
Примененные скидки
Результаты согласования
Согласование
Структура подчиненности
Электронные документы
См. также
История изменений
Продажи по заказу

Выручка и себестоимость продаж

Вариант отчета: Продажи по заказу (контекст) (Установлен дополнительный отбор) Выбрать
Сформировать Настройка...
Данные по продажам С НДС

Продажи по заказу

Параметры: Показывать продажи: Кроме продаж между организациями
Отбор: Данные по продажам: С НДС
Отбор: Заказ клиента Равно "Заказ клиента ТД-00000262 от 01.04.2010 20:12:58"

Номенклатура	Характеристика	Выручка (RUB)	Себестоимость (RUB)	Валовая прибыль (RUB)	Рентабельность %
Ботинки женские демисезонные	37, 5, , Бежевый	114 696,00	80 000,00	34 696,00	
Ботинки женские демисезонные	38, 5, , Красный	114 696,00	80 000,00	34 696,00	
Женские ботфорты	37, 6, , Черный	37 276,20	26 000,00	11 276,20	
Женские ботфорты	Размер: 38, Полнота: 7, Цвет: Черный	74 552,40	52 000,00	22 552,40	
Женские сапоги с натуральным мехом	37, 6, , Зеленый	137 635,20	96 000,00	41 635,20	
Женские сапоги с натуральным мехом	38, 5, , Красный	137 635,20	96 000,00	41 635,20	
Женские сапоги с натуральным мехом	Размер: 39, Полнота: 6, Цвет: Красный	137 635,20	96 000,00	41 635,20	
Полусапожки на шнурках	Размер: 39, Полнота: 6, Цвет: Белый	27 240,30	19 000,00	8 240,30	
Итого		781 366,50	545 000,00	236 366,50	

3.2. Если используются варианты отчетов, то пользователь не должен иметь возможность выбирать в списке вариантов отчета контекстные варианты отчета (в том числе в панели отчетов).
 3.3. В контекстных отчетах следует исключать из группировок объект анализа (контекст).

Например, группировка валовой прибыли по заказам клиентов не требуется, если отчет запущен в контексте определенного заказа клиента.

3.4. При использовании СКД имя контекстного варианта отчета должно содержать слово "Контекст".

Например, "ДвиженияСерийНоменклатурыКонтекст" отчета "ДвиженияСерийТоваров".

Варианты отчетов

Область применения: управляемое приложение.

#std671

1. Варианты отчетов следует создавать в случае, когда есть необходимость анализировать один и тот же вид (источник) данных в различных:

- группировках
- сорттировках
- отборах
- с разными наборами показателей

В этом случае создается отчет с несколькими вариантами отчетов.

Примером источников данных может являться совокупность документов и регистров сведений, которые предназначены для пользователей с одинаковым уровнем прав. Например, отчет "Выручка и себестоимость продаж":

Выбор варианта отчета

Выбрать Найти... Закрыть

Название

Анализ продаж по бизнес-регионам
Валовая прибыль по менеджерам
Валовая прибыль по партнерам
Валовая прибыль по подразделениям
Валовая прибыль по поставщикам
Валовая прибыль по сделкам
Возвраты от партнеров по месяцам
Динамика продаж
Динамика продаж по видам товаров
Объем продаж по видам товаров

Анализ результатов продаж в разрезе сделок.
Показатели: Выручка, Себестоимость, Валовая прибыль, Рентабельность.

2. Если вариант отчета является рабочим местом или используется очень часто, то такой вариант лучше оформлять отдельным отчетом. Это упростит его использование.

3. Если вариант отчета требуется включить в состав различных подсистем одного и того же раздела, то вместо одного варианта отчета следует создать свои варианты отчетов для каждой подсистемы с различными параметрами и отборами.

Например, для отчета "Валовая прибыль по поставщикам":

Правильно	Неправильно
Создать два варианта отчета: "Валовая прибыль по поставщикам (опт)" и "Валовая прибыль по поставщикам (розница)" с отборами по оптовым и розничным продажам	Создать один вариант отчета, который включить в состав подсистем "Продажи/Продажи и возвраты" и "Продажи/Розничные продажи"

4. При использовании в конфигурации Библиотеки стандартных подсистем следует обязательно заполнять описание варианта отчета. В описании отчета указывается основное назначение, состав анализируемых данных.

Поля периодов

Область применения: управляемое приложение.

#std672

1. При настройке схемы компоновки данных следует придерживаться следующих умолчаний именования полей периодов:

Поле	Путь	Синоним
Год	Периоды.Год	Период, год
Полугодие	Периоды.Полугодие	Период, полугодие
Квартал	Периоды.Квартал	Период, квартал
Месяц	Периоды.Месяц	Период, месяц
Декада	Периоды.Декада	Период, декада
Неделя	Периоды.Неделя	Период, неделя
День	Периоды.День	Период, день
Час	Периоды.Час	Период, час
Минута	Периоды.Минута	Период, минута
Секунда	Периоды.Секунда	Период, секунда

2. При настройке схемы компоновки данных следует придерживаться следующих умолчаний формата полей периодов:

Поле	Форматная строка	Пример
Год	Л=ги; ДФ='уууу "г.'"	2012 г.
Полугодие	Л=ги; ДФ='ММ.гггг "г.'"	06.2012 г.
Квартал	Л=ги; ДФ='к "квартал" гггг "г.'"	3 квартал 2012 г.
Месяц	Л=ги; ДФ='ММММ гггг "г.'"	Август 2012 г.
Декада	Л=ги; ДФ='dd.ММ.уууу "г.'"	03.08.2012 г.
Неделя	Л=ги; ДФ='dd.ММ.уууу "г.'"	03.08.2012 г.
День	Л=ги; ДФ='dd.ММ.уууу "г.'"	03.08.2012 г.

3. Для полей ПериодСекунда и Регистратор следуют придерживаться указанных ниже настроек:

- 3.1. Для поля ПериодСекунда следует установить ограничение на использование поля в качестве доступного поля выбора, отбора, группировки и упорядочивания, так указанное поле не является пользовательским;
- 3.2. Для поля Регистратор указать выражение представления «Регистратор»;
- 3.3. Для поля Регистратор указать выражение упорядочивания «ПериодСекунда возр, Регистратор возр».

Все это необходимо для корректного упорядочивания данных при расшифровке по регистратору.

Пользовательские настройки

Область применения: управляемое приложение.

#std673

Для эффективной работы пользователя с отчетом требуется определить состав быстрых и обычных пользовательских настроек. Состав настроек определяется основной целью использования отчета.

1. Общие рекомендации

1.1. Избегайте использования наименований настроек, которые могут быть истолкованы пользователями неоднозначно или быть для них непонятными.

Правильно	Неправильно
"Оплатить позже"	«Дата оплаты Меньше или равно ...»

1.2. Для параметров, без указания которых запрос системы компоновки данных не выполнится или его выполнение не имеет смысла, следует в настройках взвести флагок «Запрещать незаполненные значения» и установить режим использования «Всегда».

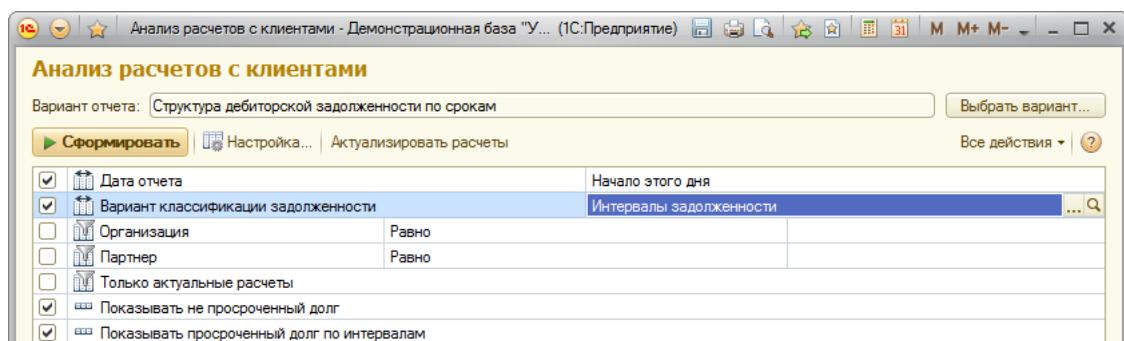
1.3. Обязательные для заполнения параметры должны заполняться наиболее вероятными значениями.

Например, период "Этот месяц" для отчета "Выручка и себестоимость продаж".

2. Быстрые пользовательские настройки

2.1. Рекомендуется делать не более 5-7 быстрых настроек.

Правильно:



Неправильно:

Анализ расчетов с клиентами - Демонстрационная база "У... (1С:Предприятие)

Вариант отчета: Структура дебиторской задолженности по срокам

Дата отчета Начало этого дня
 Вариант классификации задолженности Интервалы задолженности
 Организация Равно
 Партнер Равно
 Сегмент партнеров Равно
 Контрагент Равно
 Только актуальные расчеты
 Тип диаграммы Круговая
 Показывать не просроченный долг
 Показывать просроченный долг по интервалам

2.2. В состав быстрых пользовательских настроек следует включать только частотные настройки:

- Параметры отчета (обязательные и необязательные). Это нужно потому, что контекстное меню отчета не позволяет пользователю "на лету" применять параметры.

Валовая прибыль по менеджерам

Параметры: Период: 01.01.2011 - 31.12.2011
Показывать продажи: Кроме продаж между организациями
Данные по продажам: С НДС

Менеджер	Выручка (RUB)	Себестоимость (RUB)	Доп. расходы (RUB)	Валовая прибыль (RUB)	Рентабельность %
Федоров Борис Михайлович	1 227 440,80	801 411,35	0,24	426 029,21	34,
	498 477,39	228 190,87	1 488,42	268 798,10	53,
Яковлев Олег Константинович	681 443,81	471 575,39		209 868,42	30,
Соколов Михаил Васильевич	195 650,00	107 200,00		88 450,00	45,
Котеночкин Виктор Иванович	232 096,02	187 489,71	2,25	44 604,06	19,
Васечкин Иван Иванович	90 544,50	49 876,07		40 668,43	44,
Смирнов Олег Иванович	78 908,94	42 481,40		36 427,54	46,
Иванова Нина Юрьевна	66 102,50	29 898,50		36 204,00	54,
Симонов Иван Ильич	65 000,00	35 656,78		29 343,22	45,
Лазуренко Илья Николаевич	101 582,00	77 393,50	736,54	23 451,96	23,
		24 656,18		20 470,93	45,
		33 600,00		15 585,00	31,
		14 822,85		13 177,15	47,
			Менеджер = Лазуренко Илья Николаевич	0,30	20,
				1,98	31,
			Дополнительно...	8,50	2,
				1,80	36,8

Расшифровать...
Открыть "Менеджер = Лазуренко Илья Николаевич"

Правильно – все параметры включены в быстрые настройки:

Выручка и себестоимость продаж

Вариант отчета: Валовая прибыль по менеджерам

Период Прошлый год
 Данные по продажам С НДС
 Организация Равно
 Менеджер Равно
 Сегмент номенклатуры Равно
 Номенклатура Равно
 По номенклатуре и характеристикам

Неправильно – параметр "Данные по продажам" не включен в быстрые настройки:

The screenshot shows the 'Настройки отчета "Выручка и себестоимость продаж"' (Report settings for "Revenue and cost of sales") dialog box. In the 'Настройка' (Setting) column, the 'По номенклатуре и характеристикам' (By item and characteristics) checkbox is checked. In the 'Значение' (Value) column, the 'Данные по продажам' (Sales data) dropdown is set to 'С НДС' (With VAT). Other settings include 'Период' (Period) as 'Прошлый год' (Last year), 'Организация' (Organization) as 'Равно' (Equal), 'Менеджер' (Manager) as 'Равно' (Equal), 'Сегмент номенклатуры' (Item segment) as 'Равно' (Equal), and 'Номенклатура' (Item) as 'Равно' (Equal). The 'Настройка' column has a red border around the 'По номенклатуре и характеристикам' row.

- Отборы по соответствующим группировкам. Должны быть добавлены в быстрые пользовательские настройки для наиболее важных группировок отчета.

Правильно:

The screenshot shows the same 'Настройки отчета "Выручка и себестоимость продаж"' dialog box. The 'По номенклатуре и характеристикам' checkbox is checked. In the 'Значение' column, the 'Данные по продажам' dropdown is set to 'С НДС'. The 'Настройка' column has a green border around the 'По номенклатуре и характеристикам' row. Below the dialog, the report preview shows a table titled 'Валовая прибыль по менеджерам' (Gross profit by manager) with data for managers Fedorov and Kabel.

Неправильно:

The screenshot shows the same 'Настройки отчета "Выручка и себестоимость продаж"' dialog box. The 'По номенклатуре и характеристикам' checkbox is checked. In the 'Значение' column, the 'Данные по продажам' dropdown is set to 'С НДС'. The 'Настройка' column has a green border around the 'По номенклатуре и характеристикам' row. Below the dialog, the report preview shows a table titled 'Валовая прибыль по менеджерам' with data for manager Fedorov. The 'Номенклатура' (Item) column is highlighted with a red border.

2.3. Если в отчете несколько элементов, например, гистограмма и список, то в быстрых настройках рекомендуется предусмотреть возможность их отключения.

2.4. В быстрых настройках контекстных отчетов не следует включать отборы по полям, по которым устанавливается отбор через параметры в командах вызова отчета.

3. Обычные пользовательские настройки

3.1. Для настроек, которые являются нечастотными, лучше устанавливать режим редактирования "Обычный".

3.2. В состав обычных пользовательских настроек следует включать:

- Отборы по реквизитам объектов анализа, которые по умолчанию не выводятся в отчет, если такие требуются
Например, "Обособленное подразделение (филиал)" - реквизит поля "Организация"
- Отборы по числовым показателям отчета, если такие требуются
Например, "Сумма задолженности Больше ..." Настройки выбранных полей ("Выбранные поля")
- Настройки отборов ("Отборы")
- Настройки упорядочивания ("Сортировка")
- Настройки условного оформления ("Условное оформление")
- Настройки группировок ("Группировка")

Например, в настройках отчета "Выручка и себестоимость продаж" зеленым выделены быстрые настройки. Остальные настройки – обычные.

Настройки отчета "Выручка и себестоимость продаж"

Настройка	Значение
<input checked="" type="checkbox"/> Период	Этот месяц
<input checked="" type="checkbox"/> Данные по продажам	С НДС
<input checked="" type="checkbox"/> Выбранные поля	Выручка, Себестоимость, Доп. расходы, Валовая прибыль, Рентабельн...
<input type="checkbox"/> Организация	Равно
<input type="checkbox"/> Менеджер	Равно
<input type="checkbox"/> Сегмент номенклатуры	Равно
<input type="checkbox"/> Номенклатура	Равно
<input type="checkbox"/> Обособленное подразделение (филиал)	
<input type="checkbox"/> Отбор	
<input type="checkbox"/> Сортировка	
<input type="checkbox"/> Условное оформление	
<input type="checkbox"/> Группировка	Менеджер
<input type="checkbox"/> По номенклатуре и характеристикам	

Заголовок отчета

Область применения: управляемое приложение.

#std674

1. Заголовок отчета должен четко отражать цель анализа и содержание отчета. Заголовок должен быть узнаваемым и понятным.

2. Если используются варианты отчета, то заголовок каждого варианта всегда должен быть заполнен и соответствовать наименованию варианта отчета. Это необходимо для быстрого поиска отчета в системе по заголовку напечатанного отчета.

Правильно – в таблице отчета есть заголовок, который соответствует наименованию варианта отчета:

The screenshot shows the 'Выручка и себестоимость продаж' report window. At the top, there is a dropdown menu labeled 'Вариант отчета' with the value 'Валовая прибыль по менеджерам'. Below this, there is a 'Настройка...' button. The main area displays the report's parameters and data. The title 'Валовая прибыль по менеджерам' is highlighted with a red box. The report parameters are listed as follows:

Период	Прошлый год
Данные по продажам	С НДС
Организация	Равно
Менеджер	Равно
Сегмент номенклатуры	Равно
Номенклатура	Равно
По номенклатуре и характеристикам	

Below the parameters, the report title 'Валовая прибыль по менеджерам' is displayed again. The report data is presented in a table:

Менеджер	Выручка (RUB)	Себестоимость (RUB)	Валовая прибыль (RUB)
Федоров Борис Михайлович	1 227 440,80	801 411,35	426 029,21
	498 477,39	228 190,87	268 798,10
Яковлев Олег Константинович	681 443,81	471 575,39	209 868,42

Неправильно – в таблице отчета нет заголовка:

The screenshot shows the 1C:Enterprise software interface with the title 'Выручка и себестоимость продаж'. In the top left, there's a search bar with the text 'Валовая прибыль по менеджерам'. To its right is a button 'Выбрать вариант...' (Select variant...). Below the search bar are two buttons: 'Сформировать' (Generate) and 'Настройка...' (Settings). On the far right of the top bar are buttons for 'Все действия' (All actions) and a question mark icon.

The main panel contains several sections:

- Фильтр:** A table with checkboxes for 'Период' (Period), 'Данные по продажам' (Sales data), 'Организация' (Organization), 'Менеджер' (Manager), 'Сегмент номенклатуры' (Segment of catalog), 'Номенклатура' (Catalog), and 'По номенклатуре и характеристикам' (By catalog and characteristics).
- Параметры:** A section with the text 'Период: 01.01.2011 - 31.12.2011', 'Показывать продажи: Кроме продаж между организациями' (Show sales: Except sales between organizations), and 'Данные по продажам: С НДС' (Sales data: With VAT).
- Таблица:** A grid table with columns 'Менеджер' (Manager), 'Выручка (RUB)' (Revenue (RUB)), 'Себестоимость (RUB)' (Cost of goods sold (RUB)), and 'Валовая прибыль (RUB)' (Gross profit (RUB)). The data rows are:

Федоров Борис Михайлович	1 227 440,80	801 411,35	426 029,21
	498 477,39	228 190,87	268 798,10
Яковлев Олег Константинович	681 443,81	471 575,39	209 868,42
Соколов Михаил Васильевич	195 650,00	107 200,00	88 450,00
Котеночкин Виктор Иванович	232 096,02	187 489,71	44 604,06
Васечкин Иван Иванович	90 544,50	49 876,07	40 668,43

3. Если используются однотипные варианты отчета с различными группировками, то уточнение по группировке следует писать во множественном числе без использования разделителей (тире, скобок и т.п.), таким образом чтобы все слова названия были согласованы.

Например:

Правильно	Неправильно
Анализ расчетов с клиентами по договорам	Анализ расчетов с клиентами (по договорам)

В исключительных случаях можно использовать разделители или иные специальные знаки, например, для стандартизованной отчетности.

4. Запрещается использование названия "Основной" для варианта отчета, т.к. для новичков не понятно, что кроется за этим определением. Лучше давать осмысленные названия.

Например, для отчета "Анализ причин проигрыша сделок" варианты могут называться:

- Анализ причин проигрыша сделок по партнерам
- Анализ причин проигрыша сделок по ответственным
- Анализ причин проигрыша сделок по причинам

5. Следует избегать слова "отчет" в синониме и заголовках отчетов.

6. Если в отчете несколько элементов (например, гистограмма и список), то для каждого рекомендуется устанавливать заголовок. Общий заголовок отчета в этом случае не обязателен.

Отчеты вида "таблица", "список"

Область применения: управляемое приложение.

#std676

1. Структура отчета

Отчет должен максимально соответствовать целям и задачам анализа. Отчет будет хорошим, если глядя на него, пользователю будет понятно:

- для кого он предназначен (должность, роль пользователя)
- на какие вопросы и каким образом можно получить ответы с помощью этого отчета.

1.1. Следует проектировать отчет таким образом, чтобы количество уровней вложенности не превышало 3-х.

1.2. Вложенность группировок должна отражать взаимосвязь объектов анализа – от общего к частному.

Например, в отчете "Валовая прибыль по партнерам", видно, какая номенклатура (частное) продана определенному партнеру (общее):

Валовая прибыль по партнерам

Параметры: Период: 01.01.2011 - 31.12.2011

Показывать продажи: Кроме продаж между организациями

Данные по продажам: С НДС

Партнер Номенклатура	Характеристика	Выручка (RUB)	Себестоимость (RUB)	Доп. расходы (RUB)	Валовая прибыль (RUB)	Рентабельность, %
Дальстрой		785 554,00	493 440,64		292 113,36	37,19
+ Розничный покупатель		455 919,24	267 532,57	23,42	188 363,25	41,32
+ Алхимов А.А.		293 843,00	144 744,96		149 098,04	50,74
Вентилятор настольный, Модель 901		73 440,00	16 428,57		57 011,43	77,63
Вентилятор BINATONE ALPINE 160вт, напольный ,		60 575,00	26 550,00		34 025,00	56,17
Телевизор "JVC"		27 550,00	7 800,00		19 750,00	71,69
X-1234 BOSCH Завод бытовой техники		30 600,00	15 500,00		15 100,00	49,35
Комбайн кухонный BINATONE FP 67		33 390,00	21 000,00		12 390,00	37,11
Комбайн MOULINEX A77 4C		17 000,00	7 800,00		9 200,00	54,12
Корпус кровати		27 000,00	24 000,00		3 000,00	11,11
Чайник MOULINEX L 1,3		1 200,00	800,00		400,00	33,33
Монтаж оборудования		400,00			400,00	100,00
Белочка (конфеты)		60,50	38,00		22,50	37,19
Вентилятор ОРБИТА STERLING ЯП.		14 175,00	14 175,00			
Вентилятор настольный		8 452,50	10 653,39		-2 200,89	-26,04
+ Мастер- Холод		199 200,00	86 600,00		112 600,00	56,53
+ Иваночкин		103 650,00			103 650,00	100,00
+ Ассоль ООО		205 624,80	125 910,30	736,54	78 977,96	38,41

1.3. Группировку по периоду рекомендуется располагать в колонках таблицы с упорядочиванием по возрастанию.

Например:

Динамика оборотных средств

Параметры: Период: 01.01.2012 - 02.08.2012

Валюта отчета: RUB

Время (периоды)

Тип показателя	01.2012		03.2012		04.2012		05
	На конец периода (вал.)	Изменение (вал.)	На конец периода (вал.)	Изменение (вал.)	На конец периода (вал.)	Изменение (вал.)	
Активы минус обязательства	30 528 078,48		29 851 343,62	7 663,34	29 961 333,02	109 989,40	
Активы	30 528 078,48		29 851 343,62	7 663,34	29 961 333,02	109 989,40	
Выданные авансы	457 206,78		457 206,78		457 206,78		
Дебиторская задолженность	30 824,50		30 824,50		30 824,50		
Дебиторская задолженность (просроченная)	3 073 889,25		3 066 618,65	15 729,40	3 083 208,65	16 590,00	
Денежные средства (безналичные)	4 412 747,23		4 412 747,23		4 412 747,23		
Денежные средства (наличные)	2 159 614,03		1 503 215,83		1 524 478,83	21 263,00	
Денежные средства (у подотчетных лиц)	16 100,80		16 100,80		16 100,80		
Расчеты между организациями	618 119,62		613 119,62		632 792,62	19 673,00	
Товары на складе	19 495 037,07		19 486 971,01	-8 066,06	19 539 434,41	52 463,40	
Товары, переданные в комиссию	264 539,20		264 539,20		264 539,20		
Обязательства	(14 438 305,85)		(11 118 085,77)	15 635,00	(11 152 378,25)	-5 028,47	
Кредиторская задолженность (просроченная)	(11 969 160,62)		(9 430 999,22)	-129,80	(9 416 354,70)	14 644,53	
Полученные авансы	(1 851 025,61)		(1 433 996,81)	15 764,80	(1 835 260,81)		
Расчеты между организациями	(618 119,62)		(253 089,74)		(272 762,74)	-19 673,00	
Активы минус обязательства	16 089 772,62		18 733 257,85	23 298,34	18 436 954,77	104 960,93	

1.4. Дополнительную информацию по группировке следует выводить:

* вместе с объектом анализа, если информация уточняет свойства объекта.

Например, "Характеристика" для объекта "Номенклатура":

Ведомость по товарам организаций в ценах номенклатуры

Параметры: Период отчета: 30.07.2012 - 05.08.2012

Вид цены: Розничная

Номенклатура	Характеристика	Единица хранения	Валюта	Бытовая техника		
				Начальный остаток		Приход
				Количество	Сумма	
Кабель NYM (Севкабель) 5x35, , м		RUB				
Комбайн MOULINEX A77 4C , шт		RUB	10,000		86 940,00	
Комбайн кухонный BINATONE FP 67 , , шт		RUB	10,000		69 120,00	
Кондиционер ELEKTA С дистанционным управлением , шт		RUB	10,000			
Кондиционер ELEKTA, С ручным управлением, шт		RUB	10,000		76 550,00	
Кондиционер FIRMSTAR 12M, С дистанционным управлением, шт		RUB				
Кондиционер БК-2300, С дистанционным управлением, шт		RUB				
Кондиционер БК-2300, С ручным управлением, шт		RUB				
Корпус кровати , , шт		RUB				
Кофеварка BRAUN KF22R , , шт		RUB	10,000		54 770,00	
Кофеварка JACOBS (Австрия), , шт		RUB	10,000		48 910,00	

* в отдельной колонке, если информация, по сути, также является объектом анализа, но группировка по ней невозможна или усложнит восприятие отчета.

Например, "Валюта" в отчетах по анализу расчетов с клиентами:

Расчеты с клиентами

Параметры: Дата отчета: 02.08.2012
 Календарь: Рабочий календарь - пятидневка
 Вариант классификации задолженности: Интервалы задолженности

Общие расчеты

Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг	Граница актуальности расчетов
Интервал просроченной задолженности				Данные актуальны
EUR				Данные актуальны
RUB	6 003 844,40	5 848 417,40	2 993 371,53	Данные актуальны
USD	500,00	500,00		Данные актуальны

Детальные расчеты

Партнер	Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг
Дата платежа	Интервал просроченной задолженности			
Kikinda (Сербия)	RUB	13 740,00	13 740,00	
Алхимов А.А.	RUB	319 392,50	293 738,00	26 954,50
Альфа	RUB	253 207,00	178 689,00	90 132,00
	Не просрочено	74 518,00		90 132,00
06.12.2010	Свыше 45 дней	86 982,00	86 982,00	
03.05.2011	Свыше 45 дней	35 532,00	35 532,00	
10.05.2011	Свыше 45 дней	14 175,00	14 175,00	
18.05.2011	Свыше 45 дней	42 000,00	42 000,00	
Ассоль ООО	EUR			
Ассоль ООО	RUB	15 729,40	15 729,40	107 645,20
	Не просрочено			107 645,20
12.10.2011	Свыше 45 дней	15 729,40	15 729,40	

1.5. Показатели отчета следует располагать:

- в колонках. При этом порядок следования показателей – по важности, слева направо. В отдельных случаях показатели могут располагаться в порядке, отражающем логику вычислений (формулу).

Например, в отчете "Валовая прибыль по сделкам" показатели следуют в порядке логики вычисления:

Валовая прибыль по сделкам

Параметры: Период: 01.01.2011 - 31.12.2011
 Показывать продажи: Кроме продаж между организациями
 Данные по продажам: С НДС

Признак продаж по сделкам			Количество	Выручка (RUB)	Себестоимость (RUB)	Доп. расходы (RUB)	Валовая (F)
Партнер	Сделка	Менеджер сделки		①	②	③	④ = ① + ② + ③
Продажи по сделкам			133,000	502 204,00	308 416,00		
Мастер-Холод	Продажа холодильников	Федоров Борис Михайлович	10,000	199 200,00	86 600,00		
Балашов	Продажа вентиляторов	Федоров Борис Михайлович	20,000	28 000,00			
Алхимов А.А.	Выставочная сделка	Федоров Борис Михайлович	91,000	217 044,00	190 216,00		
Ассоль ООО	Продажа пылесосов	Федоров Борис Михайлович	12,000	57 960,00	31 600,00		
Продажи вне сделок			8 119,000	2 083 789,18	1 242 973,64		1 727,98

- в строках, если требуется сравнение нескольких показателей в динамике.

Например, в отчете "Динамика оборотных средств":

Динамика оборотных средств

Параметры: Период: 01.01.2012 - 02.08.2012
 Валюта отчета: RUB

Тип показателя	01.2012	03.2012	
Показатель	На конец периода (вал.)	Изменение (вал.)	На конец периода (вал.)
Активы минус обязательства			
Активы	30 528 078,48	29 851 343,62	7 663,34
Выданные авансы	457 206,78	457 206,78	
Дебиторская задолженность	30 824,50	30 824,50	
Дебиторская задолженность (просроченная)	3 073 889,25	3 066 618,65	15 729,40
Денежные средства (безналичные)	4 412 747,23	4 412 747,23	
Денежные средства (наличные)	2 159 614,03	1 503 215,83	
Денежные средства (у подотчетных лиц)	16 100,80	16 100,80	
Расчеты между организациями	618 119,62	613 119,62	
Товары на складе	19 495 037,07	19 486 971,01	-8 066,06
Товары, переданные на комиссию	264 539,20	264 539,20	
Обязательства	(14 438 305,85)	(11 118 085,77)	15 635,00
Кредиторская задолженность (просроченная)	(11 969 160,62)	(9 430 999,22)	-129,80
Полученные авансы	(1 851 025,61)	(1 433 996,81)	15 764,80
Расчеты между организациями	(618 119,62)	(253 089,74)	
Активы минус обязательства	16 089 772,62	18 733 257,85	23 298,34

1.6. В отчетах, где данные будут представлены в виде списка, рекомендуется колонку с порядковым номером делать первой.

1.7. Если предполагается печать отчета, то следует оптимизировать его макет под формат А4.

2. Сортировка данных

2.1. Сортировку по умолчанию следует устанавливать исходя из назначения отчета, так, чтобы наверху отображались самые важные для пользователей данные.

Правильно (сортировка по долгу клиента, сравнивать просто):

Расчеты с клиентами				
Параметры: Дата отчета: 02.08.2012 Календарь: Рабочий календарь - пятидневка Вариант классификации задолженности: Интервалы задолженности				
Общие расчеты				
Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг	Граница актуальности расчетов
Интервал просроченной задолженности				
EUR				Данные актуальны
RUB	6 003 844,40	5 848 417,40	2 993 371,53	Данные актуальны
USD	500,00	500,00		Данные актуальны
Детальные расчеты				
Партнер	Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг
Дата платежа	Интервал просроченной задолженности			
Модная обувь	RUB	1 004 386,50	1 004 386,50	799 366,50
ИнноТрейд	RUB	950 944,14	950 944,14	
Бытовая техника	RUB	825 561,01	825 561,01	32 975,10
Дальстрой	RUB	755 600,00	755 600,00	76 100,00
Алхимов А.А.	RUB	319 392,50	293 738,00	26 954,50
Бытовая техника (Владимир)	RUB	274 615,25	274 615,25	59 050,00
Никитава-частное лицо	RUB	243 500,00	243 500,00	272 000,00
Пластинформ	RUB	229 392,00	229 392,00	
Твемос	RUB	191 208,00	190 758,00	241 758,00
Пилигрим ООО	RUB	187 539,75	187 539,75	
Альфа	RUB	253 207,00	178 689,00	90 132,00
Балашов	RUB	152 096,02	152 096,02	75 150,00
Иваночкин	RUB	103 650,00	103 650,00	
Технотрейд 1	RUB	83 200,00	83 200,00	1 150,00
Электромаркет - сеть магазинов	RUB	63 315,00	63 315,00	
Стройснаб	RUB	49 004,80	49 004,80	
Домашний интерьер ООО	RUB	45 600,00	45 600,00	45 600,00
Свет	RUB	45 202,50	45 202,50	
Диваны и кровати (магазин)	RUB	34 200,00	34 200,00	10 260,00
Гришичкин	RUB	24 465,00	24 465,00	1 367,50
ООО Бюджет	RUB	38 322,00	20 317,50	81 076,50

Неправильно (сортировка по алфавиту, сравнивать сложно):

Расчеты с клиентами				
Общие расчеты				
Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг	Граница актуальности расчетов
Интервал просроченной задолженности				
EUR				Данные актуальны
RUB	6 003 844,40	5 848 417,40	2 993 371,53	Данные актуальны
USD	500,00	500,00		Данные актуальны
Детальные расчеты				
Партнер	Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг
Дата платежа	Интервал просроченной задолженности			
Kikinda (Сербия)	RUB	13 740,00	13 740,00	
Алхимов А.А.	RUB	319 392,50	293 738,00	26 954,50
Альфа	RUB	253 207,00	178 689,00	90 132,00
Ассоль ООО	EUR			
Ассоль ООО	RUB	15 729,40	15 729,40	107 645,20
База "Продовольственные товары"	RUB	1 200,00	1 200,00	
База "Электроника и бытовая техника"	RUB	2 810,91	2 810,91	
База "Электроника и бытовая техника"	USD	500,00	500,00	
Балашов	RUB	152 096,02	152 096,02	75 150,00
Беляевский-частное лицо	RUB	10 000,00	10 000,00	
Бытовая техника	RUB	825 561,01	825 561,01	32 975,10
Бытовая техника (Владимир)	RUB	274 615,25	274 615,25	59 050,00
Гришичкин	RUB	24 465,00	24 465,00	1 367,50
Дальстрой	RUB	755 600,00	755 600,00	76 100,00
Диваны и кровати (магазин)	RUB	34 200,00	34 200,00	10 260,00
Домашний интерьер ООО	RUB	45 600,00	45 600,00	45 600,00
Иваночкин	RUB	103 650,00	103 650,00	
Инвема	RUB	7 080,00	7 080,00	101 159,60
ИнноТрейд	RUB	950 944,14	950 944,14	
Кережек И.Д.	RUB			5 089,28
Коробейников Игорь Данилович	RUB			13 680,00
Магазин "Все для дома"	RUB	15 000,00	15 000,00	
Мир продуктов - сеть магазинов	RUB	2 035,00	2 035,00	
Мир продуктов (Пражская)	RUB	5 241,50	5 241,50	5 241,50
Мир продуктов (TK "Звездочка")	RUB	5 118,58	5 118,58	4 758,60

2.2. Сортировку по алфавиту (по возрастанию) следует применять в случае, когда числовые показатели являются дополнительной информацией, а целью анализа является поиск объектов учета по наименованию
Например, в отчете "Движения товаров по складам" применена сортировка по наименованию номенклатуры.

Движения товаров по складам						
Склад Номенклатура, Единица хранения	Помещение Характеристика	Количество				
		Начальный остаток	Приход	Расход	Конечный остаток	
Бытовая техника			217,000		217,000	
Вентилятор BINATONE ALPINE 160вт, напольный, шт		10,000		10,000		
Вентилятор JIPONIC (Тайв.), шт		10,000		10,000		
Вентилятор настольный, шт		10,000		10,000		
Вентилятор настольный, Модель 901, шт		10,000		10,000		
Вентилятор оконный, шт		10,000		10,000		
Вентилятор оконный, модель 900, шт		10,000		10,000		
Вентилятор ОРБИТА, STERLING, ЯП., шт		10,000		10,000		
Комбайн MOULINEX A77 4C, шт		10,000		10,000		
Комбайн кухонный BINATONE FP 67, шт		10,000		10,000		
Кондиционер ELEKTA, шт	С дистанционным управлением	10,000		10,000		
Кондиционер ELEKTA, шт	С ручным управлением	10,000		10,000		
Кофеварка BRAUN KF22R, шт		10,000		10,000		
Кофеварка JACOBS (Австрия), шт		10,000		10,000		
Миксер BINATONE HM 212,6 скор. 150вт, шт		10,000		10,000		
Миксер SOLAC мод.545, шт		10,000		10,000		
Пылесос "Омега" 1250вт, шт		20,000		20,000		
Телевизор "JVC", шт		5,000		5,000		
Телевизор "SHARP", шт		5,000		5,000		
X-1234 BOSCH Завод бытовой техники, шт		5,000		5,000		
X-6666 Атлант Холодильный комбинат, шт		5,000		5,000		
X-6666 Атлант Холодильный комбинат, шт		7,000		7,000		
X-67891 Стилон 205 Завод бытовой техники, шт		10,000		10,000		
X-77890 Стилон 101 Завод бытовой техники, шт		10,000		10,000		
Ларек "Розница"		480,000	299,000		181,000	
Магазин "Бытовая техника"		13,000	7,000		6,000	

2.3. Сортировку по периоду (по возрастанию) следует применять, когда в отчете производится анализ с учетом периода хозяйственных операций.

2.4. Сортировка в отчетах должна применяться только к видимым объектам анализа (группировкам или показателям). Пользователю должен быть понятен принцип сортировки без дополнительных объяснений.

3. Условное оформление

Оформление отдельных элементов отчета играет роль индикаторов, благодаря которым пользователь обратит внимание на важные факты. При цветовом оформлении таблиц, списков и диаграмм следует придерживаться следующих правил:

3.1. Негативные факты оформляются красным цветом (элемент стиля "НегативноеСобытие", RGB 178,34,34)

Позитивные факты оформляются зеленым цветом (элемент стиля "ПозитивноеСобытие", RGB 0,128,0)

Валовая прибыль по менеджерам						
Параметры:	Период: 01.01.2011 - 31.12.2011 Показывать продажи: Кроме продаж между организациями Данные по продажам: С НДС					
Менеджер	Выручка (RUB)	Себестоимость (RUB)	Доп. расходы (RUB)	Валовая прибыль (RUB)	Рентабельность, %	
Иванова Нина Юрьевна	66 102,50	29 898,50		36 204,00	54,77	
	498 477,39	228 190,87	1 488,42	260 798,10	53,92	← Рентабельность > 50%
Бурденко Андрей Семенович	28 000,00	14 822,85		13 177,15	47,06	
Смирнов Олег Иванович	78 908,94	42 481,40		36 427,54	46,16	
Тимофеева Елизавета Витальевна	45 127,11	24 656,18		20 470,93	45,36	
Соколов Михаил Васильевич	195 650,00	107 200,00		88 450,00	45,21	
Симонов Иван Ильич	65 000,00	35 656,78		29 343,22	45,14	
Васечкин Иван Иванович	90 544,50	49 876,07		40 668,43	44,92	
Федоров Борис Михайлович	1 227 440,80	801 411,35	0,24	426 029,21	34,71	
Покрышкина Наталья Юрьевна	15 595,00	10 640,00	3,02	4 951,98	31,75	
Орехов Вадим Геннадьевич	49 185,00	33 600,00		15 585,00	31,69	
Яковлев Олег Константинович	681 443,81	471 575,39		209 868,42	30,80	
Лазуренко Илья Николаевич	101 582,00	77 393,50	736,54	23 451,96	23,09	
Либерзон Мария Робертовна	28 925,63	23 045,33		5 880,30	20,33	
Котеночкин Виктор Иванович	232 096,02	187 489,71	2,25	44 604,06	19,22	← Рентабельность < 30%
Гладилина Вера Михайловна	31 154,00	30 345,50		808,50	2,60	
Итого	3 435 232,70	2 168 283,43	2 230,47	1 264 718,80	36,82	

3.2. Для фона ячеек с неактуальной информацией используется розовый цвет (элемент стиля "НеактуальнаяИнформация" (RGB 255,200,200)

Расчеты с клиентами

Параметры: Дата отчета: 02.08.2012

Календарь: Рабочий календарь - пятидневка

Вариант классификации задолженности: Интервалы задолженности

Общие расчеты

Валюта	Долг клиента	Долг клиента (просрочено)	Наш долг	Граница актуальности расчетов
Интервал просроченной задолженности				
EUR				Данные актуальны
Не просрочено				Данные актуальны
RUB	6 057 098,90	5 825 417,40	2 993 371,53	Данные актуальны
Не просрочено	129 772,50		2 967 717,03	Данные актуальны
Неизвестно	101 909,00		25 654,50	03.02.2011
Свыше 45 дней	5 825 417,40	5 825 417,40		Данные актуальны
USD	500,00	500,00		Данные актуальны
Свыше 45 дней	500,00	500,00		Данные актуальны

3.3. Итоговая информация оформляется жирным шрифтом (элемент стиля вида шрифт "ИтоговаяИнформация", начертание – жирный).

3.4. Для числовых полей следует включать выделение отрицательных значений в тех случаях, когда отрицательные значения однозначно указывают на ошибку в анализируемых данных или являются негативным фактом.

3.5. Помеченные на удаление данные оформляются шрифтом с зачеркиванием(элемент стиля вида шрифт "ПомеченныеНаУдалениеДанные", начертание – зачеркнутый).

4. Общие итоги

Если отображение общих итогов не имеет смысла, то его следует отключать.
Например, в отчете "Прайс-лист" для колонки "Цена" общие итоги нужно отключить.

Отчеты вида "диаграмма"

Область применения: управляемое приложение.

#std675

1. Типы диаграмм

Использование диаграмм в отчетах позволяет визуально компактно представить закономерности в данных. Это, в свою очередь, сокращает время, затрачиваемое пользователем на поиск необходимой информации.

1.1. Применять диаграммы в отчетах следует лишь в том случае, если объем данных потенциально невелик и диаграммы не будут перегружены.

1.2. Для различных целей поиска информации должны использоваться определенные типы диаграмм:

Цель поиска	Тип диаграммы	Сортировка	Условие применения
Динамика показателей	График	По возрастанию периода	Много периодов
	График по шагам		От 1-й до нескольких серий
	График с областями	По возрастанию периода	Много периодов
	График с накоплением и областями		От 2-х серий до нескольких серий
	Нормированные графики		
Распределение величины по значимым диапазонам	Вертикальные гистограммы	По возрастанию периода	Несколько периодов До 3-х серий
	Вертикальная гистограмма	Логический порядок диапазонов	Менее 10 диапазонов
	График	Логический порядок диапазонов	Более 10 диапазонов, непрерывная шкала диапазонов
Анализ структуры показателей	Круговая диаграмма	По важности значений показателя, по периоду	Анализ соотношения компонент как частей целого. Количество компонент до 7-10
	Вертикальная гистограмма	По убыванию величины показателя	Рейтинг. Количество компонент – менее 10
	Горизонтальная диаграмма	По возрастанию величины показателя	Рейтинг. Количество компонент – более 10

1.3. Следует избегать объемных диаграмм, так как они искажают информацию, отвлекают от закономерностей, которые можно обнаружить при визуальном анализе.

2. Условное оформление

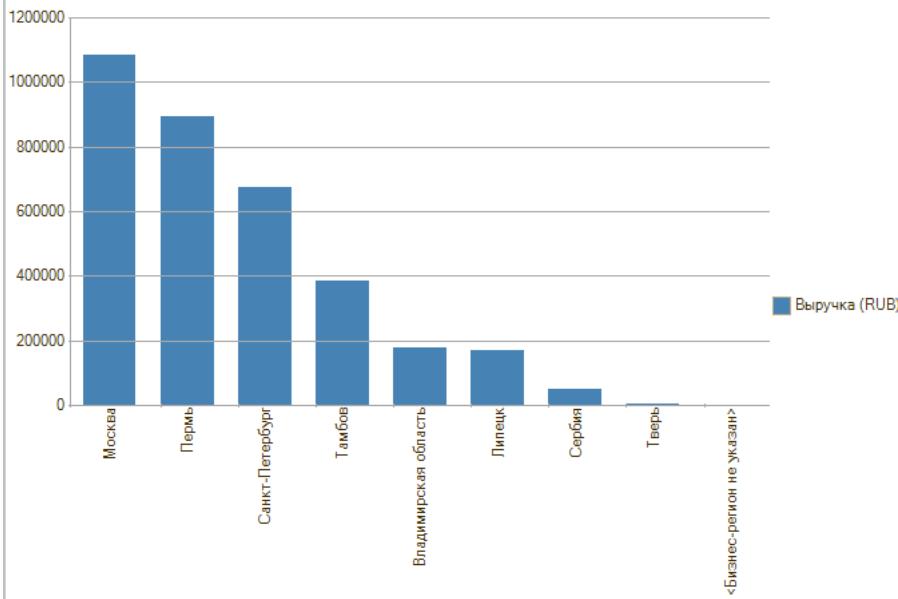
2.1. Основной цвет для гистограмм и линий графиков динамики – элемент стиля вида цвет "Диаграмма", RGB 70,130,180.

2.2. Основной цвет для прогнозных значений гистограмм и линий графиков динамики – элемент стиля вида цвет "Прогноз", RGB 199,21,133.

2.3. Следует избегать использования оттенков красного и зеленого цветов для отдельных значений (серий) в диаграммах, если значения (серии) не показывают критические (негативные) и позитивные значения.

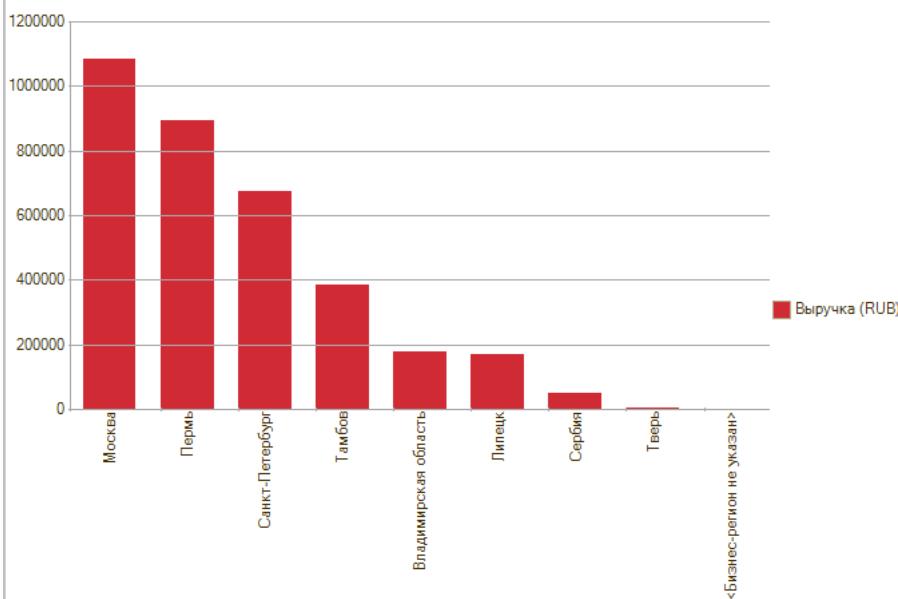
Правильно:

Анализ продаж по бизнес-регионам



Неправильно:

Анализ продаж по бизнес-регионам



Размещение большого количества команд в основном окне приложения

Область применения: управляемое приложение.

#std401

В конфигурациях может возникнуть проблема размещения большого числа команд в панели действий. Как правило, это касается команд формирования отчетов. Использование для большого числа команд панели действий не оптимально. Для решения данной проблемы предлагается следующая методика.

Выполняется анализ всех команд отчетов, отнесенных к определенной подсистеме (сложенными подсистемами);

- Для большинства из них снимается видимость по умолчанию;
- Для некоторых, наиболее востребованных отчетов, видимость по умолчанию оставляется;
- При необходимости, настройка видимости команд по умолчанию выполняется «по ролям», но для самого «полноправного» пользователя число команд, видимых по умолчанию в панели действий, не должно превышать 6-9 команд.

Для подсистемы верхнего уровня реализуется общая форма с названием, включающим в той или иной форме имя подсистемы, например, «Все отчеты по зарплате». В такой форме размещаются все команды той или иной подсистемы;

- В форме все команды реализуются как гиперссылки;
- В такой форме может быть реализована «отсыпка» к настойке панели действий;
- Большой размер рабочей области позволяет разместить также команды смежных областей, т.е. реализовать подобие раздела «см. также».

Реализуется общая команда, например, **ВсеОтчетыПоЗарплате**:

- Код команды представляет собой просто открытие общей формы

ОткрытьФорму("ОбщаяФорма. ВсеОтчетыПоЗарплате", ,
ПараметрыВыполненияКоманды.Источник,
ПараметрыВыполненияКоманды.Уникальность,
ПараметрыВыполненияКоманды.Окно)

- Название такой команды подчеркивает то, что она охватывает все отчеты. Например, «Все отчеты по зарплате».
- Команда включается в состав нашей подсистемы верхнего уровня.

- Команда относится к одной из групп Панели навигации

На рисунке ниже приведен пример внешнего вида основного окна приложения при выполнении такой команды как «Все отчеты по зарплате»



Рабочий стол

Область применения: управляемое приложение.

#std578

Рабочий стол – личный заботливый помощник пользователя. Каждый рабочий день начинается с «общения» с ним.

Рабочий стол должен вводить пользователя в курс дел, отвечая на его вопросы, например:

- Что нужно сделать сегодня?
- Что появилось нового?
- На что следует обратить внимание?
- Каково состояние важных для меня сведений?

Что размещать на рабочем столе

При определении состава форм, доступных на рабочем столе следует исходить из набора ролей конфигурации. Каждой роли может соответствовать одна или более форм рабочего стола. Т.к. пользователи, как правило, относятся к нескольким ролям, то рабочий стол будет состоять из комбинации форм этих ролей.

При проектировании рабочего стола важно учитывать:

- Рабочий стол – первое, что увидит пользователь при запуске программы, поэтому его следует разрабатывать особенно тщательно.
Для командной панели рабочего стола допускается снятие флагка «Автозаполнение» и их ручное заполнение.
К формам рабочего стола нужно относиться, как к микро рабочим местам. С одной стороны они не должны быть перегружены, но с другой, предоставлять богатые возможности.
Например, список задач может поддерживать drag-and-drop, чтобы при перетаскивании в него заказа покупателя система сразу предлагала создать новую задачу для коллеги со ссылкой на этот заказ.
- Содержимое рабочего стола формируется автоматически в соответствии с ролями. Как правило, пользователь выполняет несколько ролей.
Например, менеджер по продажам – занимается продажами, ведет личный учет времени, исполняет общекомпанийные поручения и приказы, выполняет анализ рынка, ведет исследования для развития своего направления.

Что нужно учитывать при разработке

- Рабочий стол является обязательным разделом и не может быть отключен.
- Проектируйте формы рабочего стола под существенно меньший размер, чем обычные формы.
- Не рекомендуется размещать на формах рабочего стола списки с горизонтальной полосой прокрутки.
- Не рекомендуется использовать в формах рабочего стола меню «Все действия».
- По возможности, не вставляйте команду вызова справки в формы рабочего стола.
- В командных панелях форм рабочего стола рекомендуется делать минимально необходимый набор команд.
- В командной панели списков не рекомендуется делать команды поиска и отмены поиска.
- Формы для рабочего стола следует проектировать «с запасом» - делая их по умолчанию не видимыми, чтобы пользователи имели возможность настроить рабочий стол под себя.
- В командных панелях динамических списков рекомендуется размещать команду обновления.
- Состав форм рабочего стола следует проектировать таким образом, чтобы рабочий стол реального пользователя, с учетом ролей, содержал по умолчанию от 3-х до 6-ти форм. Иначе рабочий стол будет смотреться вырожденным или перегруженным.
- В списках на рабочем столе по умолчанию должно быть видно не менее 3,5 строк.

Панель действий на рабочем столе

- Панель действий на рабочем столе рекомендуется делать только в том случае, если можно достоверно выделить частотные и не контекстные команды:
 - Команды, которыми будут пользоваться часто;
 - Команды, для выполнения которых не нужно переключаться в другой раздел.
- При этом нужно следить, чтобы ПД не стала непонятной из-за разнородности команд, попавших в нее из разных разделов.

Рабочее место

Область применения: управляемое приложение.

#std615

Рабочее место — это форма для выполнения какой-то одной бизнес-операции или группы взаимосвязанных бизнес-операций.

Примеры: Приемка, Отгрузка, Заказы клиентов.

Следовательно, рабочее место должно удовлетворять двум основным требованиям:

- представлять необходимый набор команд (создание новых данных, анализ имеющихся, выполнение других контекстных действий);
- обеспечивать пользователя актуальной информацией, необходимой для правильного и быстрого выполнения бизнес-операций.

Примеры: Статусы выполнения, сроки исполнения, приоритетность.

Другие требования к рабочему месту

- Рабочее место должно быть интуитивно понятно сотруднику, который знает свою работу, но не знаком с программой.
- Своим оформлением рабочее место должно «говорить», для какого специалиста и для решения каких задач оно предназначено.
- Предусмотрите на рабочем месте необходимый состав инструментов для работы специалиста.
- Акцентируйте внимание пользователя на важной информации с помощью цвета.
- Проследите, чтобы визуально рабочее место не казалось перегруженным.
- Рабочее место должно содержать достаточный минимум сведений, необходимых для качественного выполнения пользователем своих задач.
- Самые важные сведения располагайте сверху.
- Размещайте инструменты в логической последовательности решения задач.
- Продумайте оптимальное выполнение действий: без необходимости переключения в другие интерфейсы.
- Предусмотрите, в случае необходимости, переход к косвенным задачам (рабочим местам).
- Названия рабочим местам следует давать по смыслу решаемых ими задач.

Примеры: «Администрирование», «Расчет заработной платы», «Регистрация корреспонденции».

- В названиях рабочих мест не рекомендуется использовать словосочетание «рабочее место».
- Не рекомендуется размещать внедренную справку на рабочем месте.

Внедренная справка — справка в форме, оформленная в виде подробных комментариев к полям, либо в виде отдельного поля, содержащего справку к форме. Для реализации внедренной справки следует указывать подсказку элемента формы с режимом отображения «Снизу», используя стиль "ПоясняющийТекст" RGB (128, 122, 89). В качестве исключения, если нет возможности добиться желаемого визуального эффекта с помощью подсказки элемента, можно использовать декорацию «Надпись».

Где размещать команду перехода к рабочему месту

- В группе «Важное» панели навигации, если решаемые задачи частотны. При этом команда будет отображаться жирным шрифтом.
- В блоке «См. также» панели навигации, если в контексте конкретной подсистемы рабочее место имеет второстепенное, вспомогательное значение и используется не часто.

Формы списков

Область применения: управляемое приложение.

#std616

- Состав и порядок колонок должен определяться прогнозируемым сценарием заполнения.
- Отбирайте и группируйте колонки в несколько этажей так, чтобы можно было обойтись без горизонтальной полосы прокрутки.
При этом нужно учитывать, что в многоэтажных списках труднее воспринимать информацию при беглом чтении.
- Фиксируйте колонки там, где это уместно.
Например, «№» и «Номенклатура» в списках документов.

При конфигурировании можно вставлять в таблицы динамических списков колонки «с запасом», делая их по умолчанию невидимыми, чтобы пользователь со временем мог оптимально настроить список по себе. Эту рекомендацию не следует применять для табличных частей.

- Рекомендуется по умолчанию делать активной колонкой ту, поиск по которой будет наиболее вероятным.
Например, для табличной части списка товаров это будет колонка «Товар».
- Не рекомендуется использовать режим выделения строки – строка.

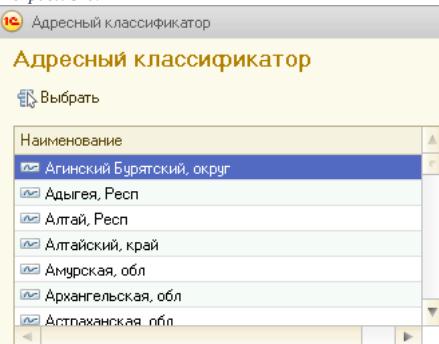
Списки с одной колонкой

Область применения: управляемое приложение.

#std617

- Не рекомендуется делать шапку.
- Если заголовка нет, то визуально пустой список ничем не будет отличаться от многострочного поля ввода. Это нужно учитывать, чтобы не ввести пользователя в заблуждение.
- Не рекомендуется горизонтальная разлиновка и чередование строк.

Неправильно:



Правильно:

Адресный классификатор

Адресный классификатор

Выбрать

Список регионов и областей:

- Агинский Бурятский, округ
- Алтай, Респ
- Амурская, обл
- Астраханская, обл
- Башкортостан, Респ
- Брянская, обл
- Владимирская, обл
- Вологодская, обл

Размеры списков

Область применения: управляемое приложение.

#std618

- Высота списков по умолчанию может быть подобрана с учетом типового количества строк. При этом нужно исходить из практики. Например, для списка товаров в УНФ это не более 5-10 штук. Для списка документов заказов покупателя 3-5 строк мало, более 20 уже много т.к. такое количество все равно не охватить одним взглядом.
- Не обязательно стремиться к одинаковой высоте списков, например, в различных документах. Они должны быть узнаваемыми, но точного соблюдения геометрии не требуется.
- Высота строки управляемого списка при обычном шрифте – 19 pt.
- Если нужно понять какой ширины должна быть колонка или поле и известно типовое количество символов в значении, используйте формулу:
Ширина колонки = Количество символов/1,25

Минимальные размеры колонок для таблиц управляемой формы

Колонка	Количество символов (с пробелами)	Ширина
Банк	20	16
Банковский счет	20	16
БИК	9	7
Валюта	3	3
Год	4	4
Дата (со временем)	19	13
Договор	24	19
Индекс	6	5
ИНН физ. лица	12	10
ИНН юр.лица	10	8
Код ИФНС	4	4
Комментарий	20	16
Контрагент	20	16
КПП	9	7
Месяц	8	7
Номенклатура	20	16
Номер	11	9
Номер ГТД	30	23
ОГРН	13	10
OKATO	11	9
Описание	20	16
Организация	20	16
Ответственный	15	12
Подразделение	20	16

Какую высоту строки использовать

- Одинарную, если значения в ячейках короткие и список выглядит компактно и не перегружено.
- Двойную, если значения в ячейках длинные и важно их видеть полностью.

Быстрые отборы в списках

Область применения: управляемое приложение.

#std581

1. Когда использовать

1.1 Если список используется часто и просматриваются типовые поисковые «запросы».

Например, по контрагенту, за период и т.д.

1.2 Если список подразумевает большое количество объектов, что создаст трудности для самостоятельного поиска (идентификации) объекта пользователем.

1.3. При работе со списком частотной является операция выбора/поиска.

2. Как оформлять быстрые отборы

2.1 Поля ввода или флажки рекомендуется размещать перед списком сверху, слева или справа, в зависимости от состава отборов и их смысла. Допускается размещать внизу списка, если отбор полезный, но не частотный.

Например, «Показывать выполненные задачи» в списке «Мои задачи».

2.2. Быстрые отборы сверху следует располагать над командной панелью списка.

2.3 У полей необязательных отборов должна быть кнопка очистки или должна быть одна кнопка очистки всех полей быстрого отбора.

2.4 Быстрые отборы следует оформлять без кнопки открытия

2.5 Если поля быстрых отборов по умолчанию сильно растягиваются, то для них следует устанавливать фиксированные размеры (при условии, что хорошие размеры можно заведомо подобрать)

Многоэтажные списки

Область применения: управляемое приложение.

#std604

- Когда данные важно просматривать без прокрутки.
- Если в списке много колонок, например, более 10.
- Если для просмотра данных пользователям часто придется использовать горизонтальную полосу прокрутки.
- При проектировании учитывайте, что в многоэтажных списках визуальный поиск данных сложнее, чем в одноэтажных.

Команды, размещаемые во "Всех действиях"

Область применения: управляемое приложение.

#std606

- Рекомендуется использовать умолчание платформы.
- Для отдельных списков допускается индивидуальная настройка командной панели. При этом непосредственно в командной панели формы лучше располагать небольшое количество наиболее частотных команд (не более 10), а остальные размещать во «Все действия».

Заголовки списков

Область применения: управляемое приложение.

#std607

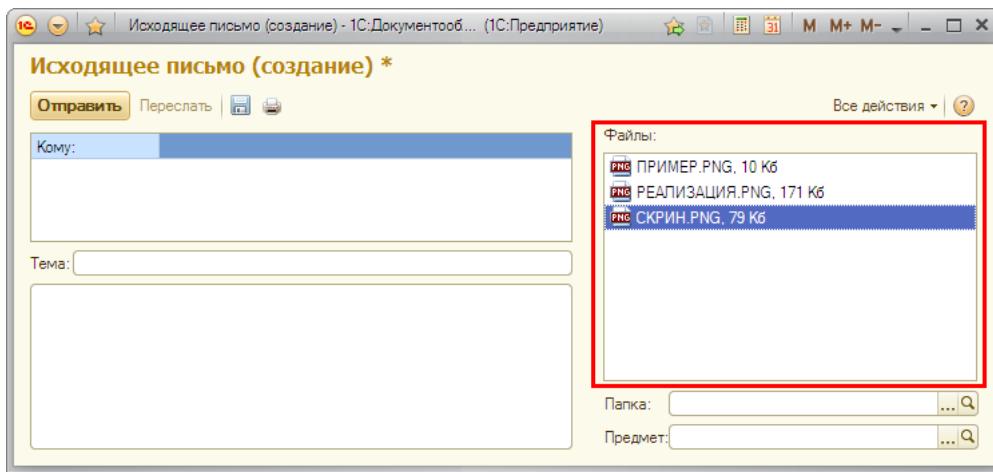
1. Оформление заголовка списка

1.1. Если список имеет командную панель, то его заголовок выводится в виде группы с рамкой "линия".

Например, список "Выданные авансы":

1.2. Если список не имеет командной панели, то его заголовок выводится в положении "верх".

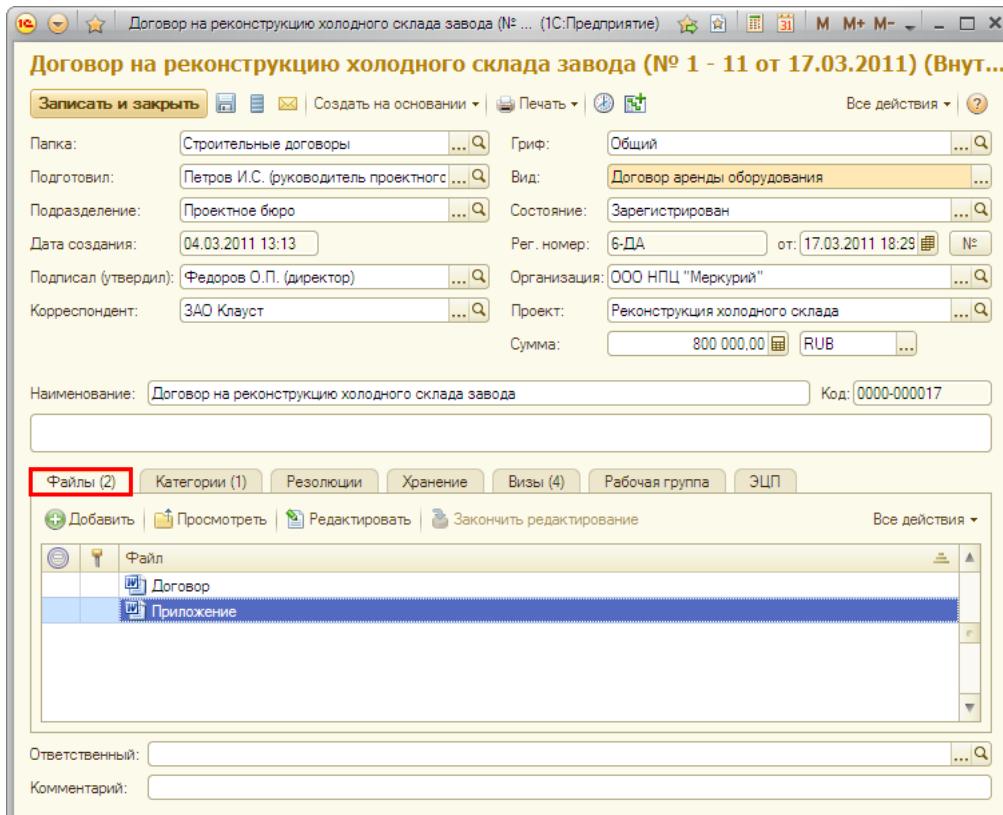
Например, список "Файлы" в карточке письма:



2. Отражение количества элементов в списке

2.1. Количество элементов следует показывать только в тех случаях, когда это уместно и помогает пользователю решить какую-либо задачу.

Например, в карточке внутреннего документа количество в заголовке списка файлов показывает, с каким числом файлов нужно ознакомиться пользователю:



2.2. Если в заголовке конкретного списка принято выводить количество элементов, то это нужно делать во всех формах программы, где этот список используется и решает аналогичные задачи.

Например, количество в заголовке списка файлов выводится во всех видах документов: входящих, исходящих, внутренних.

2.3. Количество элементов в списке выводится после заголовка, в круглых скобках.

Например, число товаров в списке "Товары" документа "Отбор (размещение) товаров":

Отбор (размещение) товаров (создание) - Демонстрационная (1С:Предприятие)

Провести и закрыть | Провести | Создать на основании | Печать | Все действия | ?

Распоряжение: Расходный ордер на товары 0000000023 от 02.04.2010 12:15:00

Номер: ЦУ-00000002 от: 11.03.2011 17:47:01 | Вид операции: Отбор | Статус: Выполнено

Склад: Центральный склад | Помещение: Коммерческие товары

Зона отгрузки: Отгрузка | Рабочий участок:

Исполнитель:

Товары (19)

+ Добавить | Заполнить | Все действия

N	Ячейка	Номенклатура	Характеристика	Серия	Упаковка	К отбору	Отбрано
1	T2-8	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	2,000	2,000
2	T2-7	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	3,000	3,000
3	T2-6	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	3,000	3,000
4	T2-5	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	3,000	3,000
5	T2-4	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	3,000	3,000
6	T2-3	Телевизор "SHAR...	<характеристики ...	<серии не указыв...	шт (1 шт)	1,000	1,000

Ответственный: Федоров Борис Михайлович | ... |

2.4. В пустом списке количество элементов выводить не требуется.

Хорошо

Плохо

N	Номенклатура	Количество

3. Ситуации, в которых заголовок списка не выводится

3.1. Если на форме только один список. В качестве заголовка выступает название формы.
Например, журнал "Кассовые документы":

Кассовые документы

Организация: Конфетпром ООО | ... |

+ Создать | Найти... | Все действия | ?

Дата	Номер	Приход	Расход	Контрагент	Вид
05.04.2012 16:59:54	0000-000001		100.00		Взн
06.04.2012 17:05:59	0000-000002		200.00	ООО "Старт"	Опл
06.04.2012 17:07:14	0000-000002	3 000.00		ИП Иванов Н.В.	Опл

3.2. Если на форме два списка, связанных общим контекстом. При этом назначение списков должно быть интуитивно понятно и не требовать дополнительного пояснения. В качестве общего заголовка выступает название формы.

Например, дерево папок файлов и список файлов в папке:

The screenshot shows a software interface for managing files. On the left, there is a tree view of a folder structure under 'Юзабилити'. The 'Книги и полезные мат...' folder is expanded, showing sub-folders like 'Все для ю-тестирования' and 'Полезные доклады ю...'. On the right, a list view displays various files with columns for 'Наименование' (Name) and 'Описание' (Description). Some files have small icons next to their names.

3.3. Если список выводится на странице (вкладке). В качестве заголовка выступает название страницы (вкладки).
Например, вкладка "Товары" в документе "Счет на оплату покупателю".

The screenshot shows a document editing interface for an invoice. The title bar says 'Счет на оплату покупателю 0000-000001 от 13.04.2012 9:42:28 *'. The main area contains fields for 'Номер' (Number), 'Контрагент' (Counterparty), 'Договор' (Contract), 'Адрес доставки' (Delivery address), and a tabbed section for 'Товары (1)' (Products). A table below lists one product: 'Стол' (Table) at a quantity of 10.000, unit price of 3 000.00, total sum of 30 000.00, and VAT of 18%. At the bottom, there are fields for 'Ответственный' (Responsible person) and 'Комментарий' (Comment).

В остальных случаях заголовок списка рекомендуется показывать.

Колонки с флагжками

Область применения: управляемое приложение.

#std627

- Избегайте длинных заголовков т.к. колонка при этом выглядит некрасиво, занимает много места: флагок прижат влево и вся колонка пустая.
- По возможности, заменяйте названия таких колонок на понятные картинки с хорошей подсказкой или используйте сокращения. Необходимо помнить, что большое количество картинок замедляет открытие формы.

Группировки в списках

Область применения: управляемое приложение.

#std609

- Группировку в списке стоит использовать, если есть уверенность в том, что она будет способствовать визуальному поиску нужных данных
- Если группировок просматривается несколько, то их можно вынести как настройку в область быстрого отбора.
- Следует помнить, что группировки могут сильно затормозить работу (см. [Ограничения при использовании динамических списков](#))

См. также: [Быстрые отборы в списках](#)

Команда "Создать" в журналах документов

Область применения: управляемое приложение.

#std582

1. Для создания документов в некоторых журналах рекомендуется использовать подменю «Создать» вместо стандартной команды «Создать». Вариант создания через подменю подходит, если:

- в журнале отражается небольшое количество видов документов (до 6)
- журнал часто используется в работе

2. В подменю размещаются команды для открытия конкретных типов документов.

3. Картинку следует добавлять только для подменю «Создать», для каждого типа команды в подменю ее использовать нет необходимости.

Например, в журнале «Кассовые документы» доступны команды создания «Приходного кассового ордера», «Расходного кассового ордера» и т.д.



4. В подменю «Создать» рекомендуется сортировать команды по частоте использования, а не в алфавитном порядке.

5. При использовании подменю «Создать» поведение клавиши Insert не меняется – при нажатии на нее открывается стандартная форма «Выбор типа документа».

Пояснение невозможности заполнения ячеек в табличных частях

Область применения: управляемое приложение.

#std610

В случае, если ячейка в табличной части не нуждается в заполнении в том или ином состоянии, необходимо сообщать пользователю о причине. Текст рекомендуется оформлять так:

- угловые скобки
- цвет текста: «ТекстЗапрещеннойЯчейки» (RGB: 192,192,192)
- с маленькой буквы

Например, в случае если номенклатура не использует характеристики, в ячейке «Характеристика» выводится сообщение в угловых скобках: <характеристики не используются>

N	N по док.	Номенклатура поставщика	Номенклатура	Характеристика	Количество
1			Ассорти (конфеты)	<характеристики не используются>	20,000
2			Барбос (конфеты)	<характеристики не используются>	20,000
3			Белошка (конфеты)	<характеристики не используются>	20,000
4			Грильяж (конфеты)	<характеристики не используются>	20,000
5			Крупа "Геркулес"	<характеристики не используются>	20,000
6			Крупа гречневая (весовая)	<характеристики не используются>	20,000
7			Крупа гречневая (упак.)	<характеристики не используются>	20,000
8			Крупа манная	<характеристики не используются>	20,000
9			Масло "Кремлевское"	<характеристики не используются>	20,000
10			Масло вологодское	<характеристики не используются>	20,000
11			Масло деревенское	<характеристики не используются>	20,000
12			Мишса (конфеты)	<характеристики не используются>	20,000
13			Молоко "Домик в деревне" 1.5%	<характеристики не используются>	20,000

Акцентирование внимания на просроченных или критичных состояниях

Область применения: управляемое приложение.

#std584

При необходимости привлечь внимание, например к истекающему состоянию, сроку выполнения операции и т.п., рекомендуется выделять только тот элемент, который поможет пользователю понять в чем причина привлечения внимания и что делать дальше.

Цвет текста: "ПросроченныеДанные" (RGB 178; 34; 34)

Например, в случае если истекает срок действия Соглашения с поставщиком, необходимо выделять только ячейку состояния «действует» и ячейку «действует по».

Номер	Дата	Наименование	Текущее состояние	Валюта	Действует с	Действует по
	15.03.2010	Комиссионная поставка бытовой техники (База "Электр...")	Действует	RUB	15.03.2010	
		Поставка кухонной техники (База "Электроника и БТ")	Действует	RUB	31.03.2010	
	25.01.2010	Поставка кухонной техники (Дом быта)	Действует	RUB	25.01.2010	
	20.02.2010	Поставка мебели (База "Мебель")	Действует	RUB	20.02.2010	
	05.03.2010	Поставка мебели (Мебельная фабрика)	Действует	RUB	05.03.2010	15.05.2010
	25.03.2010	Поставка обуви (ЭКИП ООО)	Действует	RUB	25.03.2010	25.05.2011
	15.01.2010	Поставка продуктов	Действует	RUB	15.01.2010	
	08.02.2010	Поставка техники (Пабан)	Действует	RUB	08.02.2010	

Групповые обработки в списках

Область применения: управляемое приложение.

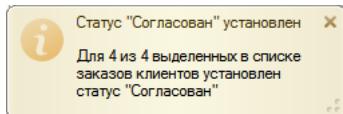
#std653

Решаемые проблемы:

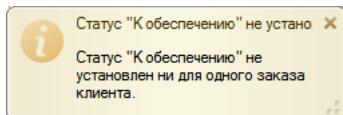
- Различные отклики программы на одни и те же действия пользователя
- Пользователь остается без отклика программы на свое действие

1. При использовании групповых обработок необходимо сообщать пользователю о количестве удачно выполненных операций.

Например, при изменении статуса у выделенных заказов клиентов программа сообщает пользователю, что статус был успешно изменен у всех заказов:

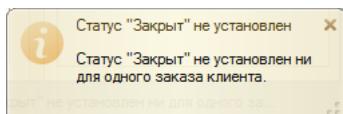


В случае возникновения ошибок программа сообщает пользователю:



2. Нельзя оставлять пользователя без отклика в случае возникновения ошибок. В случае, если часть или все объекты групповой обработки выполнены неудачно, необходимо сообщить об этом пользователю.

Например, пользователь пробует изменить статус у Заказов клиентов, которые находятся в статусе "Закрыт" на статус "Закрыт". Система сообщает пользователю, что документы уже находятся в статусе "Закрыт" и в изменении статуса нет необходимости.

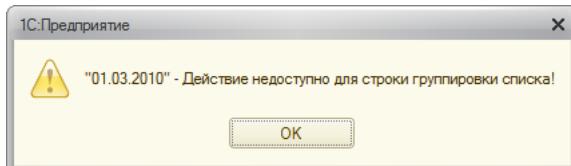


3. Групповые обработки в списках с группировкой списка

Ордера на отгрузку товаров				
Дата отгрузки	Приоритет	Номер	Документ	
01.03.2010		ЦУ-00000019	Расходный ордер на товары	К проверке
01.03.201...				
31.03.2010		ЦУ-00000003	Расходный ордер на товары	К проверке
31.03.201...				
21.04.201...		ЦУ-00000002	Расходный ордер на товары	К проверке
10.02.2011				
10.02.201...		ЦУ-00000024	Расходный ордер на товары	К отгрузке

При реализации групповых обработок при установленной группировке списка необходимо соблюдать следующие требования:

- При выделении нескольких строк, в том числе строк группировки, в момент групповой обработки игнорировать строки группировок. Обработка применяется только к выделенным строкам.
- При выделении только строки группировки и применении групповой обработки выдавать платформенное сообщение:



Сообщения пользователю

#std585

Область применения: управляемое приложение.

1. Общие рекомендации

1.1. Сообщения должны быть достаточно информативными и содержательными. Сообщения составляются в форме безличного предложения: не употребляются местоимения "Вы", "Вас" и пр.

Правильно

"Недостаточно прав для выполнения обработки"

Неправильно

"У Вас недостаточно прав для выполнения обработки"

1.2. Сообщения не должны содержать восклицательных знаков и повелительных тонов, за исключением сообщений, предупреждающих об опасных или критических действиях.

Правильно

"Внимание. Загрузка базы может привести к потере всех данных!"

Неправильно

"Внимание! Загрузка базы может привести к потере всех данных!"
Это сообщение не акцентирует внимание пользователя на том, что последствия могут критичны

"Выберите элемент, а не группу"

"Выберите элемент, а не группу!"

Это сообщение несет негативный настрой, а также заставляет пользователя чувствовать себя глупо

1.3. Сообщение должно быть написано текстом, понятным пользователю. В частности, оно не должно содержать технической терминологии (терминологии разработчика). Подобную терминологию допускается использовать только в сообщениях, предназначенных для администратора. Например, если пользователь указал в качестве кассы отправителя и кассы получателя одно и то же значение, то выводится сообщение: "Касса отправителя равна кассе получателя". В реальности касса не может быть равна другой – она может быть одной и той же.

Правильно

"Касса отправителя и касса получателя должны различаться"

Неправильно

"Касса отправителя равна кассе получателя"

1.4. Сообщения, указывающие на ошибку, должны выводиться в момент возникновения ошибки.

Например, в отчете, сдаваемом в Пенсионный фонд, указываются регистрационные номера сотрудников. Эти номера заполняются в карточке сотрудника.

Проверку правильности заполнения этого номера нужно делать в момент его ввода в карточке, а не в момент формирования отчета.

1.5. В тексте сообщений необходимо избегать двусмысленности. Категорически запрещается постановка вопроса с содержанием частиц "не" и "бы". Подобные вопросы не способны дать однозначного ответа.

Правильно

"Удалить файл?"

Неправильно

"Не хотели бы вы удалить этот файл?" (Да / Нет)

"Переместить или удалить этот файл?" (Да / Нет)

1.6. В текстах сообщений не следует использовать сокращения и аббревиатуры. Исключением являются только общеупотребительные и соответствующие целевой аудитории сокращения и аббревиатуры.

Например, сокращения "НДС", "МСФО" понятны пользователям, а "К." (коэффициент) – нет.

1.7. Если текст сообщения занимает несколько строк, рекомендуется составлять его таким образом, чтобы строки оканчивались на логическую паузу или завершались знаком препинания.

Правильно

Не рекомендуется изменять значение ставки налога, если она уже используется в справочниках или документах.

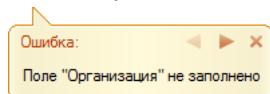
Неправильно

Не рекомендуется изменять значение ставки налога, если она уже используется в справочниках или документах.

1.8. Следует руководствоваться принципом, что в процедурах и функциях, выполняемых в транзакции, нельзя выполнять каких-либо взаимодействий с пользователем – выводить предупреждения и вопросы.

1.9. Сообщения, оповещения, предупреждения или состояние не следует использовать в случае, если нужно вывести протокол, отчет или подробности о результатах выполненной операции. Для этого следует предусмотреть специальную форму.

2. Сообщение об ошибках в форме



Сообщения об ошибках в форме выводятся в панели сообщений об ошибках справа:

Счет на оплату покупателю (создание)

Провести и закрыть | Провести | Создать на основании | Печать | Все действия | ?

Номер: от: 30.04.2013 0:00:00 | Контрагент: | ... |

Организация: | ... | Договор: | ... | Новый договор |

Склад: | ... | Цена включает НДС |

Подразделение: | ... | Поле "Организация" не заполнено |

Адрес доставки:

Реквизиты для оплаты

Получатель: | ... | Банковский счет: | ... |

Товары | Возвратная тара | Услуги |

+ Добавить | ... | Подбор | Изменить | Все действия |

N	Номенклатура	Количество	Цена с НДС	Сумма с НДС	% НДС

Всего: 0.00 руб. НДС (в т.ч.): 0.00 руб.

Ответственный: Админ | ... |

Комментарий: | ... |

[Как вывести ответственных лиц организации в документах](#) | [Как создать счет на оплату](#) | Все

Сообщения

- ! Поле "Организация" не заполнено
- ! Поле "Контрагент" не заполнено
- ! Поле "Получатель" не заполнено
- ! Поле "Банковский счет" не заполнено

2.1. В панели сообщений следует выводить только сообщения об ошибках. Сообщения об успешном завершении операции в панели выводить запрещается.

2.2. Сообщение об ошибке по возможности должно быть привязано к полю, которое породило ошибку или позволяет исправить ее. Если сообщение нельзя привязать к тому или иному полю, то в тексте сообщения следует явно указать, что пользователю нужно сделать для того, чтобы устраниТЬ проблему.

2.3. Сообщения об ошибках должны отвечать на 3 вопроса:

- Что произошло?
- Почему это произошло?
- Что делать дальше?

Ответов на эти вопросы должно хватить пользователю, чтобы принять решение о дальнейших действиях, а также не повторить данной ошибки в будущем.

Например, пользователь пытается изменить статус документа, помеченного на удаление.

Система выдает сообщение: "Документ "Коммерческое предложение <номер> не проведен. Статус не изменен"

Сообщение создает новый вопрос для пользователя: почему не изменен статус? В тексте сообщения отсутствует информация о том, что документ помечен на удаление, и что именно это является причиной возникновения ошибки.

Правильно

"Невозможно изменить статус: документ
"Коммерческое предложение <номер> помечен на
удаление"

Неправильно

"Документ "Коммерческое предложение <номер> не
проведен. Статус не изменен"

2.4. Следует учитывать, что в панели сообщений текст сообщения автоматически форматируется, а строки переносятся, сохраняя при этом свои пропорции. Поэтому текст таких сообщений не нужно разбивать на строки.

См. также: [Перенос выражений](#)

2.5. Сообщение должно быть как можно более кратким и понятным.

Если же поле заполнено, но неправильно, то сообщение должно быть более подробным.

Например, "Указанного количества товара <название товара> нет на складе. Доступно: <количество доступно> <единица измерения>".

2.6. Текст сообщения об ошибке должен содержать побудительную часть, призывающую пользователя совершить действия по исправлению ошибки.

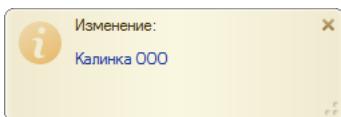
Правильно

"Укажите хотя бы одну систему налогообложения"

Неправильно

"Необходимо указать хотя бы одну систему
налогообложения"

3. Оповещение



3.1. Рекомендуется использовать оповещение для информирования пользователя о произошедших событиях без прерывания основной работы. Пользователю не обязательно реагировать на оповещение, оно выдается для информации. Оповещение сообщает о том, что запрошенная операция (запись элемента справочника или проведение документа) выполнена.

3.2. Оповещения рекомендуется делать с гиперссылками на соответствующие объекты.

3.3. Текст и пояснение оповещения рекомендуется составлять так, чтобы они целиком помещались в окне оповещения с размерами "по умолчанию":

- Текст – 36 знаков;
- Пояснение – около 100 знаков (с учетом переноса на три строки).

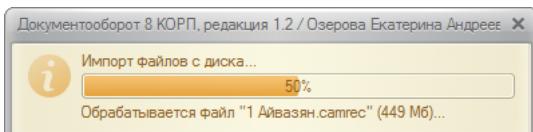
4. Состояние

4.1. Рекомендуется использовать состояние для информирования о выполнении длительных процессов (занимающих более 10 секунд), чтобы у пользователей не сложилось впечатление о том, что программа "зависла". Выводите состояние:

- перед началом выполнения ("Выполняется расчет. Пожалуйста, подождите...")
- в процессе выполнения (если есть возможность)
- при завершении ("Расчет выполнен")

4.2. Состояние используется для длительных операций, состоящих из некоторого числа более мелких операций.

Например, при переносе файлов с жесткого диска в информационную базу можно выводить состояние для каждого переносимого файла:



См. также: [Длительные операции, Запись событий в историю работы пользователя](#)

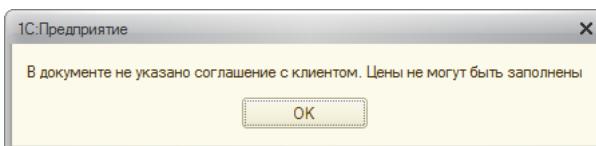
5. Предупреждение

5.1. Предупреждение рекомендуется использовать только в тех случаях, когда необходимо прервать работу пользователя, чтобы он ознакомился с некоторой информацией. При этом для возобновления работы пользователю не нужно принимать решений.

5.2. Не следует использовать предупреждение для информирования о начале выполнения длительных обработок. Такое сообщения следует выдавать непосредственно в форме обработки.

5.3. В тексте предупреждения следует привести законченные пояснения о последующих действиях и их последствиях.

Например, пользователь пытается заполнить цены товаров в табличной части "Товары" документа «Заказ клиента». Программа выдает предупреждение о невозможности выполнения этого действия:



6. Вопросы в сообщениях

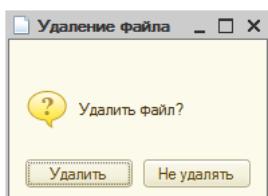
6.1. Рекомендуется использовать в тех случаях, когда необходимо, чтобы пользователь принял решение, например, о продолжении начатой операции.

6.2. Вопросы рекомендуется выдавать перед выполнением:

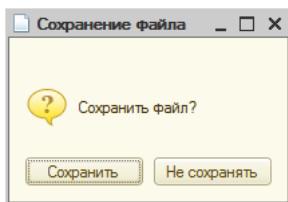
- действий, результаты которых отменить потом невозможно
- потенциально опасных для данных пользователя действий
- массовой обработки информационной базы
- длительных процедур

6.3. Ответами на вопрос должны выступать глаголы, обозначающие последующие действия.

Например:



6.4. В сообщениях-вопросах кнопкой по умолчанию должна являться та кнопка, выбор которой наиболее безопасен для данных пользователей.
Например, при сохранении файла кнопкой по умолчанию является "Сохранить":



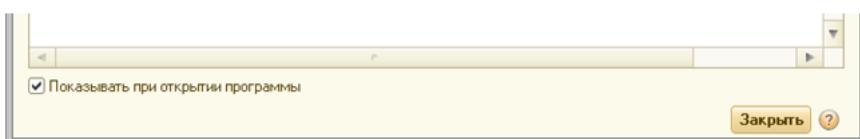
ВАЖНО: Чаще всего в ОС Windows кнопкой по умолчанию является первая (самая левая) кнопка. Прибегать к изменению кнопки по умолчанию рекомендуется только в исключительных ситуациях.

Окна на старте

Область применения: управляемое приложение.

#std586

Если содержание окна на старте не обязательно для ознакомления пользователям, то необходимо предусмотреть по умолчанию возведенный флаг «Показывать при открытии программы». Флаг рекомендуется располагать под содержанием окна.



Формы

Область применения: управляемое приложение.

#std619

Форма должна быть оптимизирована под клавиатурный ввод:

- Логичный сценарий переходов по клавише Tab.
- Последний переход должен быть на кнопку завершающую сценарий.
- Связанные объекты должны быть расположены так, чтобы переходы по клавише Tab осуществлялся последовательно.
Например, переход с выбора партнера на выбор контрагента.

См. Также: [Общие рекомендации компоновки форм](#)

Компоновка форм

Область применения: управляемое приложение.

#std597

Компоновка форм (8.3)

1. Общие рекомендации

1.1. Один и тот же реквизит в разных формах рекомендуется размещать в одинаковом месте.

Например, поле «Контрагент» располагается в левой верхней части всех форм, в которых присутствует.

1.2. В шапке формы документа размещаются реквизиты, используемые для его правильной регистрации в программе. Сведения, относящиеся к содержательной части документа, итоговые и второстепенные реквизиты в шапке не указываются.

1.3. Шапка документа может состоять из одной, двух или трех колонок. Число и состав колонок определяются в зависимости от количества реквизитов документа.

2. Оформление шапки, состоящей из двух колонок

2.1. В левой колонке рекомендуется размещать основные сведения о документе, а в правой – второстепенные.

Левая колонка	Правая колонка
Важные для заполнения поля	Вспомогательные, малозначительные поля
Поля, заполняемые пользователем	Поля, заполняемые автоматически, значение которых требуется только проанализировать
Обязательные для заполнения поля	Не обязательные для заполнения поля
Поля, значения которых заполняются по данным, приходящим извне	Поля, значения которых пользователь устанавливает самостоятельно по внутренним данным

2.2. Реквизиты рекомендуется распределять по колонкам в следующем порядке:

Левая колонка	Правая колонка
1. Номер, дата	1. Организация

2. Вид операции	2. Банковский счет (организации)
3. Контрагент	3. Подразделение
4. Банковский счет (контрагента)	4. Склад
5. Договор	5. Касса
6. Контактное лицо	6. Регистрация в ИФНС

2.3. При наличии в форме нескольких взаимозависимых полей, их следует располагать последовательно – одно под другим.
Например, поле «Подразделение» располагается сразу под полем «Организация»

2.4. Если в форме есть поле «Вид операции», то под ним следует размещать реквизиты, состав которых зависит от выбора вида операции, за исключением реквизитов, отражаемых в содержательной части формы.

3. Примеры оформления конкретных групп полей

Левая колонка	Правая колонка
Номер, дата + Вид операции Номер: <input type="text"/> от: <input type="text"/> : : <input type="button" value="..."/> <input type="button" value="🔍"/> Вид операции: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>	Организация + Банковский счет (организации) Организация: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Банковский счет: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>
Контрагент + Договор Контрагент: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Договор: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>	Организация + Подразделение + Склад Организация: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Подразделение: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Склад: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>
Контрагент + Банковский счет (контрагента) Контрагент: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Банковский счет: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>	Организация + Подразделение Организация: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Подразделение: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>
Контрагент + Контактное лицо Контрагент: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Контактное лицо: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>	Организация + Касса Организация: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Касса: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>
	Организация + Регистрация в ИФНС Организация: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/> Регистрация в ИФНС: <input type="text"/> <input type="button" value="..."/> <input type="button" value="🔍"/>

Командная панель формы

Область применения: управляемое приложение.

#std620

См. также: [Командная панель документа \(8.3\)](#)

В командной панели формы кнопка по умолчанию должна быть самой левой (первой)

Порядок полей

Область применения: управляемое приложение.

#std682

- Кнопкам рекомендуется устанавливать свойство «ПропускатьПриВводе».
- Для поля, с которого обычно начинается ввод, рекомендуется установить свойство «АктивизироватьПоУмолчанию».

Размеры

Область применения: управляемое приложение.

#std681

- Установку размера формы и/или ее элементов рекомендуется выполнять только в особых случаях, например, для декораций. В остальных случаях рекомендуется использовать умолчание платформы.
- Для форм с панелями со страницами рекомендуется подбирать ширину так, чтобы одновременно были видны все закладки страниц.
- Ширину колонок форм рекомендуется изменять только в тех случаях, если колонки содержат простой набор хорошо масштабируемых полей и групп, например, два поля списка в левой и правой колонке.
- Для форм объектов рекомендуется использовать авто ширину колонок (установлено по умолчанию).

Группы элементов формы.

Область применения: управляемое приложение.

#std621

Применение

- Если элементы формы можно объединить в группу с общим признаком.
В группу «Доверенность» входят элементы формы: номер, дата выдачи, кем выдана, кому выдана.
- Для облегчения визуального восприятия формы за счет объединения элементов формы.

Количество элементов в группе

- Нет необходимости делать группы, содержащие один элемент.
- Рекомендуется размещать в группах 4-9 элементов. В случае если элементов более 9 необходимо разбить их на две и более группы.

Оформление заголовка группы

- Заголовок может использоваться в качестве расшифровки или пояснения, если из названий реквизитов не достаточно понятно их назначение.
- Не рекомендуется использование заголовка, если группа одна и расположена на странице панели (вкладке). В этом случае заголовком группы является заголовок самой страницы.

См. также

- [Элементы форм: требования по локализации](#)

Поля "Ответственный" и "Комментарий"

#std587

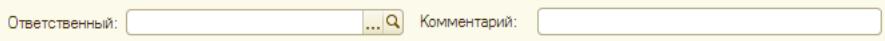
Область применения: управляемое приложение.

См. также: [Поля "Ответственный" и "Комментарий" \(8.3\)](#)

Поля "Ответственный" и "Комментарий", используемые в формах документов, необходимо располагать и оформлять в соответствии со следующими рекомендациями.

1. Расположение группы полей "Ответственный" и "Комментарий"

Группу полей "Ответственный" и "Комментарий" следует располагать в самом низу формы, друг за другом:



2. Оформление поля «Ответственный»

- Поле не растягивается на всю ширину формы
- Для поля подбирается такая ширина, чтобы в него полностью помещались типичные выбираемые значения. На максимально возможные значения ориентироваться не следует.



3. Оформление поля «Комментарий»

- Заголовок располагается слева от поля
- Поле растягивается по горизонтали на всю ширину формы
- Поле не растягивается по вертикали

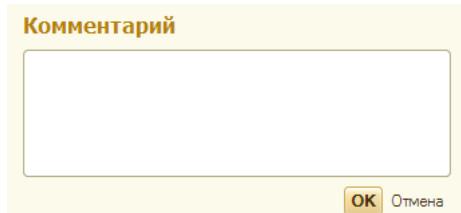
Как правило, поле "Комментарий" является многострочным (свойство "МногострочныйРежим" = Да). В случае, если в поле "Комментарий" всегда будет вводиться небольшое количество информации, его рекомендуется делать однострочным (свойство "МногострочныйРежим" = Авто).

Решение о том, каким должно быть поле – однострочным или многострочным – принимается разработчиком исходя из прикладных требований.

3.1. Оформление многострочного поля "Комментарий":



- Высота поля – две строки
- Используется кнопка выбора, по которой открывается окно редактора многострочного текста (блокирующее):



3.2. Оформление однострочного поля "Комментарий":



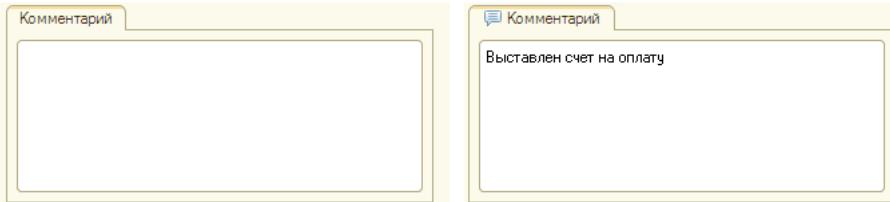
- Высота поля – одна строка

- Кнопка выбора не используется

Если поле «Комментарий» в типовых случаях будет большим, то допускается вынести его на отдельную вкладку

4. Оформление вкладки «Комментарий»

- Заголовок вкладки – «Комментарий»
- На вкладке размещается поле с текстом комментария, которое растягивается по вертикали и по горизонтали. Заголовок поля не отражается
- Если поле с комментарием заполнено, то в заголовке вкладки отражается картинка «Комментарий»



См. также: [Реквизит "Комментарий" у документов](#)

Группа полей "Наименование", "Код", "Полное наименование", "Входит в группу"

Область применения: управляемое приложение.

#std588

Поля, содержащие Наименование объекта (одно или несколько), Код и Группу, в которую входит объект, рекомендуется объединять вместе.

В качестве заголовка для Группы рекомендуется использовать понятный заголовок, например «Входит в группу». Заголовки остальных полей не следует менять без крайней необходимости, чтобы они могли соответствовать синонимам соответствующих реквизитов объектов метаданных.

Расположение полей:

- Группа полей размещается в верхней части формы
- Первыми располагаются поля «Наименование» и «Код» - горизонтально в одну строку. Поле «Код» следует за полем «Наименование»
- При наличии поля «Полное наименование» оно размещается под полями «Наименование» и «Код»
- Поле «Входит в группу» рекомендуется располагать последним

Рекомендуется пропускать поле «Код» при вводе с клавиатуры.

Примеры расположения полей:

Наименование:	<input type="text"/>	Код:	<input type="text"/>
Входит в группу:	<input type="button" value="..."/> <input type="button" value="🔍"/>		
Наименование:	<input type="text"/>	Код:	<input type="text"/>
Полное наименование:	<input type="text"/>		
Входит в группу:	<input type="button" value="..."/> <input type="button" value="🔍"/>		

Панель навигации вспомогательного окна

Область применения: управляемое приложение.

#std611

- Если нужно отображать в форме объекта (документа, справочника) какую-то дополнительную информацию (не хранящуюся в самом объекте), связанные списки, отчеты, и т.п., то это следует реализовывать с помощью команд, отображаемых в панели навигации вспомогательного окна, а не путем помещения этой информации в саму форму (в том числе и на отдельные закладки).

Лучше пользователю давать единообразные решения, так как иначе получается, что часть связанной информации он получает через панель навигации, а часть через переключение закладок.

Наличие лишних элементов может замедлять открытие формы.

- Для частотных и важных форм рекомендуется вручную настраивать видимость команд в ПН, отбирая только команды перехода к важным для пользователя сведениям.
- Рекомендуется размещать в ПН не более 10 команд.
- Не следует размещать в панели команды перехода к другим формам для редактирования реквизитов объекта. Открывать форму элемента из панели навигации окна другой формы элемента недопустимо.

Список, открываемый из панели навигации формы объекта

Область применения: управляемое приложение.

#std684

- Список, открываемый из панели навигации формы объекта, следует отражать с отбором по этому объекту.

Например, в списке «Договоры» справочника «Контрагент» отражаются договоры только по конкретному контрагенту, например, по ООО «Контрагент»

- В списке не следует выводить название объекта (в виде надписи или поля ввода в режиме "просмотр"), т.к. оно отображается в области системных команд и в панели навигации

Правильно	Неправильно
-----------	-------------

3. В списке не следует отражать колонку с названием объекта, т.к. для каждой строки это название будет одинаковым.

Например, в списке «Договоры» справочника «Контрагент» колонка Контрагент не выводится.

4. В названии команды, отражаемой в панели навигации, не следует указывать название объекта, т.к. его можно определить из контекста.

Правильно

Банковские счета
Договоры
Ответственные лица

Неправильно

Банковские счета контрагента
Договоры контрагента
Ответственные лица организаций

Выбор: блокирующая форма или независимая

Область применения: управляемое приложение.

#std589

Блокирующая

- Работа с формой выполняется «за один заход», без необходимости переключения в другие формы.
- На форме небольшое количество элементов, например, менее 5.

Независимая

- Если при работе с формой может потребоваться открытие других самостоятельных форм.
- Если пользователям может потребоваться сравнение двух и более объектов.
- На форме много элементов, больше 9.

См. также

- [Блокирующее или независимое открытие форм объектов](#)

Формы выбора

Область применения: управляемое приложение.

#std612

- Рекомендуется не создавать формы выбора, а использовать генерируемые платформой по умолчанию.
- Если, в соответствии с прикладной логикой, в форме выбора нужно предусмотреть особенный состав команд или колонок, то можно создать форму выбора, следуя рекомендациям
 - В командной панели формы рекомендуется размещать минимально необходимый набор команд для выбора, создания нового и поиска/отбора.
 - В часто используемых формах выбора из больших наборов данных рекомендуется делать область быстрого отбора/поиска.
 - Состав колонок следует оптимизировать для быстрого визуального поиска данных

См. также: [Быстрые отборы в списках](#)

Формы пошаговых помощников (мастеров)

#std696

Область применения: управляемое приложение.

Пошаговые помощники (мастера) используются для последовательного, контекстно-зависимого ввода данных. В помощнике весь процесс ввода данных разбивается на несколько отдельных этапов.

[Примеры пошаговых помощников: помощник ввода нового партнера, помощник приема сотрудника на работу, помощник создания опроса, помощник обновления системы и т.д.](#)

1. Расположение и оформление кнопок

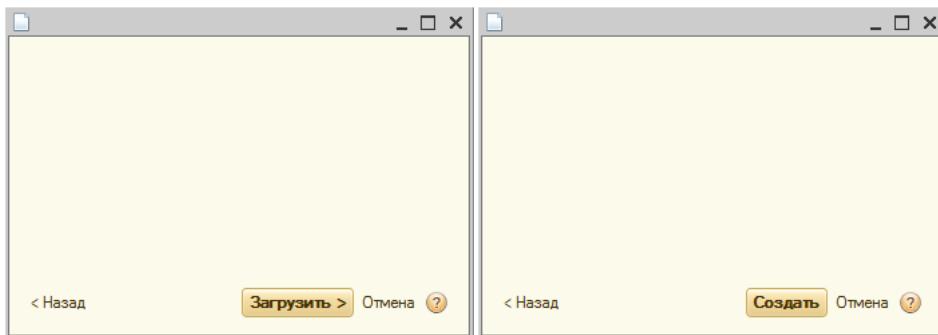
1.1. Кнопки, которые используются в форме пошагового помощника, оформляются в виде командной панели, расположенной в нижней части формы. Подобное размещение командной панели характерно именно для форм помощников.



1.2. На каждом этапе помощника одна из кнопок должна являться кнопкой по умолчанию и отличаться от других кнопок визуально – желтым цветом фона и жирным шрифтом.

Для разных экранов помощника кнопка по умолчанию может называться по-разному.

Как правило, кнопка по умолчанию – это команда, наиболее часто используется на конкретном экране.



2. Состав кнопок

2.1. Состав кнопок определяется для каждого помощника индивидуально и зависит от задач, которые решаются с помощью него.

Командная панель помощника может включать:

- Кнопку по умолчанию:
 - о "Далее"
 - о Кнопка, подтверждающая ввод данных
 - о "Закрыть"
- "Назад"
- "Отмена"
- "Справка"

2.2. Кнопка «Далее»

Используется для перехода к следующему этапу помощника. В название кнопки добавляется угловая скобка ">".



2.3. Кнопка "Назад" (при наличии)

Используется для перехода к предыдущему этапу помощника. В название кнопки добавляется угловая скобка "<".

В командной панели кнопка "Назад" всегда располагается в крайнем левом углу. На первый экран помощника кнопка "Назад" не добавляется.



2.4. Кнопка, подтверждающая ввод данных

Используется для безусловного применения всех действий, выполненных в помощнике.

В качестве названия для этой кнопки следует использовать глагол, характеризующий действие, которое будет выполнено при нажатии на нее.



В случае если такое название подобрать сложно, можно использовать слово "Готово".



Название "OK" лучше не использовать, т.к. пользователю будет сложнее попасть в кнопку из-за небольшого размера, а также понять, какое действие подразумевается под "OK".

Если кнопка, подтверждающая ввод данных, расположена на одном из промежуточных экранов помощника, то в название кнопки добавляется угловая скобка ">".

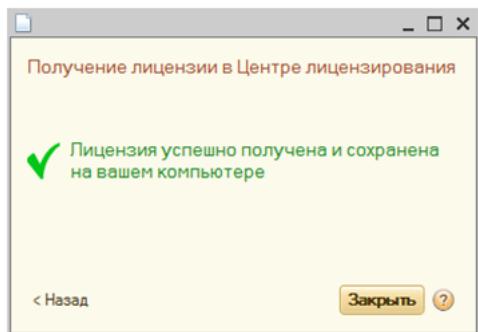
В случае расположения кнопки на последнем экране помощника, угловая скобка ">" не добавляется.



2.5. Кнопка "Закрыть"

После нажатия на кнопку, подтверждающую ввод данных, может быть показана информация, содержащая результаты выполнения операции (как положительные, так и отрицательные). Если эта информация выводится справочно, и от пользователя не требуется выполнять никаких дополнительных действий, то в качестве кнопки по умолчанию используется кнопка "Закрыть".

При нажатии на эту кнопку форма помощника закрывается.



2.6. Кнопка "Отмена" (при наличии)

Позволяет пользователю отказаться от ввода данных и закрыть форму помощника. В случае если пользователь ввел какие-либо данные в форме, то при нажатии на кнопку "Отмена" следует выдавать сообщение с вопросом об их сохранении.

[Отмена](#)

В командной панели кнопка "Отмена" располагается между кнопкой, подтверждающей ввод данных, и кнопкой "Справка".

[Далее >](#) [Отмена](#) [?](#)

2.7. Кнопка "Справка" (при наличии)

Используется для отражения справочной информации по вводу данных в помощнике. Если подобной информации не предусмотрено, то кнопку "Справка" отражать не нужно.

[?](#)

В командной панели кнопка "Справка" всегда располагается крайней справа.

[Далее >](#) [Отмена](#) [?](#)

Взаимосвязанные поля

Область применения: управляемое приложение.

#std634

Взаимосвязанные поля – когда заполнение одного поля зависит от значения, выбранного в другом поле.

Например, заполнение поля «Договор» зависит от того, какой будет выбран «Контрагент»; заполнение поля «Подразделение» – от того, к какой «Организации» оно относится и т.п.

Пользователю необходимо визуально показать, какое из полей следует заполнить первым.

Для этого поле, заполнение которого зависит от другого поля, следует оформлять недоступным до того момента, пока не будет заполнено основное поле.

Например, поле «Договор» следует выделять как недоступное до того момента, пока поле «Контрагент» не будет заполнено.

Контрагент:	<input type="text"/> ... <input type="button" value=""/>
Договор:	<input type="text"/> ... <input type="button" value=""/>

Командные панели табличных частей

Область применения: управляемое приложение.

#std695

Этот стандарт распространяется на табличные части форм объектов и не распространяется на формы списков.

Командная панель табличной части формы может содержать команды, применяемые:

- к табличной части в целом
- к отдельным строкам табличной части (одной или нескольким)
- к одной строке табличной части

1. Команды, применяемые к табличной части в целом

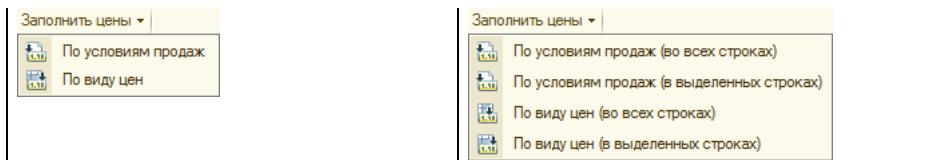
скоро появится

2. Команды, применяемые к отдельным строкам табличной части

2.1. Команды, влияющие на заполнение отдельных строк, должны применяться исключительно к выделенным строкам.

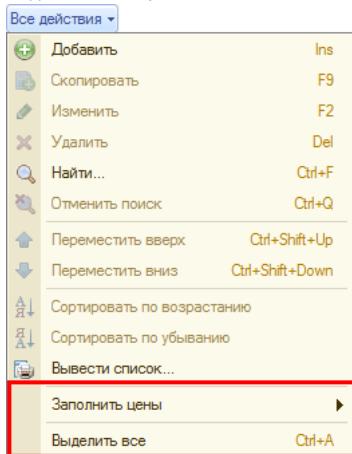
Например, следует использовать одну команду "Заполнить по виду цен" вместо двух – "Заполнить по виду цен во всех строках" и "Заполнить по виду цен в выделенных строках":

Правильно	Неправильно



2.2. При добавлении команды, влияющей на заполнение отдельных строк, в подменю "Все действия" следует добавлять команду "Выделить все". Это нужно делать в связи с тем, что многие пользователи не знакомы с горячими клавишами Ctrl+A.

Например: команда "Выделить все" добавлена в подменю "Все действия" для того, чтобы можно было быстро заполнить цены не только в выделенных строках, но и во всех

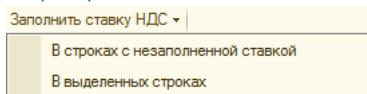


2.3. Допускается добавлять команды, которые будут действовать не на выделенные строки, а на строки, отобранные по специальному алгоритму. Это следует делать только для частотных операций, по которым пользователю сложно самостоятельно выделить строки.

Например, команда для операции автоматического заполнения видов цен только по тем строкам, в которых вид цен не заполнен.

2.4. Если в одной командной панели есть схожие команды, одна из которых действует на выделенные строки, а другая – на все, то из названия команд должно быть понятно, на какие именно строки они влияют.

Например:



3. Команды, применяемые только к одной строке
скоро появится

См. также:

[Групповые обработки в списках](#)
[Подменю](#)

Оформление элементов

Область применения: управляемое приложение.

#std624

См. также: [Компоновка форм \(8.3\)](#)

Названия одних и тех же объектов, их расположение должны быть одинаковые для всех форм.

Пример: Номер, дата документа, клиент/поставщик, контрагент, соглашение всегда находятся в одном и том же месте, вне зависимости от документа.

Переход к форме с дополнительными реквизитами

#std602

Область применения: управляемое приложение.

См. также: [Компоновка форм \(8.3\)](#)

Для оформления перехода к форме, в которую вынесена часть реквизитов из основной формы, рекомендуется использовать гиперссылку.

Например, для перехода:

- к форме «Цена и валюта» из формы «Поступление товаров и услуг»
- к форме «Реквизиты платежа в бюджет» из формы «Платежное поручение»

Платежное поручение (создание) *

Записать и закрыть | Создать на основании | Всё действия | ?

Номер: от: : : Организация: Конфетпром ...
Получатель: ... Банковский счет: Основной счет ...
Счет получателя: ...

Сумма платежа: 0.00 Вид платежа: Не указывать ...
Ставка НДС: ... Очередность платежа: 6
Сумма НДС: 0.00

Перечисление в бюджет

Реквизиты платежа: 18210101013011000110; 70401000000; ТП; МС.04.2011; 0; 0; НС; Статус: 01

Назначение платежа:

Ответственный: Петрова Марианна Александровна
Комментарий:

Реквизиты платежей в бюджет * (1С:Предприятие)

Реквизиты платежа в бюджет *

Вид перечисления: Налоговый платеж ...

Статус составителя: 01 - налогоплательщик (плательщик сборов) - юридическое лицо ...
Тип платежа: НС - уплата налога или сбора ...
Код ОКАТО: 70401000000 ...

КБК
Разряды 1-3: 4-13: 14-17: 18-20:
182 ... 1010101301 ... 1000 110

Основание платежа: ТП - платежи текущего года ...
Налоговый период: МС - месячный платеж ... Год: 2011 ... Месяц: 4 ...
Налоговый период, за который производится уплата налога (сбора)

Номер документа: 0
При основании "ТП" проставляется "0"
Дата документа: 0 - значение не указывается ...
Дата подписи налогоплательщиком декларации, представленной в налоговый орган

OK Отмена ?

Перед гиперссылкой выводится подпись, в которой указывается заголовок открываемой формы:

Цена и валюта: <Тип цен: не указан>; Валюта: руб.; Цена включает НДС

Реквизиты платежа: <КБК: не указан>; <ОКАТО: не указан>; ТП; МС.03.2011; НС; Статус: 01

Текст гиперссылки информирует пользователя о том, какие реквизиты можно заполнить в открываемой форме.

Что указывается в тексте гиперссылки

- Реквизиты, заполняемые в открываемой форме чаще остальных
- Реквизиты, значения которых влияют на порядок заполнения основной формы
- Реквизиты, для которых установлена обязательность заполнения

Как оформляется текст гиперссылки

- Реквизиты перечисляются через точку с запятой и разделяются пробелом
Например, «НС» вместо «Тип платежа: НС»
- Если значение реквизита не заполнено, то вместо него отражается текст «не указан». При этом заголовок реквизита и текст «не указан» отражаются в угловых скобках
Например, <Тип цен: не указан>, <КБК: не указан>
- Для реквизита в виде флажка выводится его заголовок, если флаг установлен, и заголовок с частицей «не», если флаг не установлен
Например, «Цена включает НДС» (флаг установлен) и «Цена не включает НДС» (флаг не установлен)

См. также: [Выбор: кнопка или гиперссылка](#)

Выбор: кнопка или гиперссылка

Область применения: управляемое приложение.

#std599

Кнопка

- Хорошо подходит для обозначения одиночной, самостоятельной операции (обработка, расчет).
- Используется, как правило, для запуска некоторого процесса, выполнения действий по изменению данных информационной базы. Необратимые изменения (например, очистка табличных частей) всегда должны быть оформлены кнопками.

- Название у кнопки должно содержать глагол, отвечающий на вопрос «Что сделать?» «Найти», «Рассчитать», «Записать», «Заполнить», «Выбрать», «Подобрать». Вместо глаголов можно использовать отглагольные существительные.
- Кнопки следует размещать в непосредственной близости от объектов, на которые они оказывают воздействие.

Гиперссылка

- Используются для перехода к другой форме, которая отображает связанные данные (другие объекты);
- Из названия гиперссылки должно быть понятно, что именно откроется в новом окне, например, «цены номенклатуры», «паспортные данные».

Картинки (иконки) в названии команд

Область применения: управляемое приложение.

#std625

Использовать картинки (иконки) необходимо в тех случаях, когда есть уверенность четкой трактовки функции, которая будет выполнена. А также в случаях, когда необходимо выделить команду на фоне других. В иных случаях команду достаточно вывести в виде текста.

Если в командной панели только одна команда, то она обязательно должна иметь картинку.

Только картинка

Картинки можно использовать без подписи в случаях, если это стандартные функции, такие как: сохранить, удалить, редактировать, скопировать и т.п., а также часто используемые функции.

Важно: Избегайте использования иконок, метафоры которых не способны отразить вызываемой функции.

Частотные кнопки

Область применения: управляемое приложение.

#std590

- Для оформления кнопок с часто используемыми действиями (частотных кнопок) рекомендуется использовать текст совместно с картинкой.

Картинка, размещенная на кнопке, также может использоваться в строке табличной части для дополнительной идентификации типа данных, к которому применима данная команда.

Кнопки, не предназначенные для решения основных задач

- Пункты меню, не предназначенные для решения основных задач, а исполняющих команды, предназначенные для специализированных целей или особой группы пользователей, рекомендуется помещать в подменю.

Итоги в документах

Область применения: управляемое приложение.

#std613

См. также: [Итоги в документах \(8.3\)](#)

Итоги в документах можно размещать несколькими способами:

- в подвале колонок таблиц
- отдельными полями после таблиц
- в специальной области итогов в нижней части формы документа

Оформление итогов в подвалах таблиц рекомендуется применять в следующем случае:

- если в таблице нет горизонтальной полосы прокрутки и итоговая колонка всегда видна
- если итог должен быть явным образом визуально связан именно с этой колонкой таблицы

В остальных случаях лучше размещать итоговые данные в специальной области итогов в нижней части форм документов.

В области итогов не рекомендуется размещать элементы, не относящиеся к итоговым данным. Их следует располагать до или после области итогов.
Например, ссылку на счет-фактуру следует размещать до области итогов, а группу полей «Ответственный» и «Комментарий» - после.

Для оформления области итогов можно использовать следующие способы:

- Объединение итоговых показателей в группы. В этом варианте поля итогов рекомендуется оформлять как надписи

Заказ клиента ТД00-000266 от 05.04.2010 13:04:33 - Демонстрационная база "Управление торговлей", редакция 11 / Ф... (1С:Предприятие)

Заказ клиента ТД00-000266 от 05.04.2010 13:04:33

Провести и закрыть | Провести | Создать на основании | Печать | ЭД | Зачет оплаты | Изменить | Все действия | ?

Номер: ТД00-000266 от: 05.04.2010 13:04:33 Статус: Согласован ... Приоритет: Средний ...
Клиент: Ахимов А.А. Организация: Торговый дом "Комплексный" ...
Контрагент: Ахимов А.А. Склад: Центральный склад ...
Соглашение: Оптовые продажи Операция: Реализация клиенту Валюта: RUB

Товары (5) Этапы оплаты (1) Дополнительно Комментарий

+ Добавить | X | Подобрать товары | Цены | Скидки (наценки) | Дата отгрузки | Отмена строк | Все действия

N	Номенклатура	С...	Характеристика	Количество	Упаковка, Ед. изм.	Вид цены	Цена
1	Молоко "Домик...		<характеристики не используются>	24,000	шт	Оптовая	
2	Молоко "Домик...		<характеристики не используются>	24,000	шт	Оптовая	
3	Молоко "Домик...		<характеристики не используются>	24,000	шт	Оптовая	
4	Молоко "Останк...		<характеристики не используются>	12,000	шт	Оптовая	
5	Молоко "Останк...		<характеристики не используются>	12,000	шт	Оптовая	

Итоговая сумма (RUB): 4 936,00 | Итоговая скидка (наценка): Всего: 200,00 4% | Этапы оплаты (1): Аванс: 4 936,00 100% | Расчеты (RUB): Долг: 0,00 0%
НДС: 448,73 | Авто: 200,00 4% | Предоплата: 0,00 0%
Отменено: 0,00 | Ручная: 0,00 0% | Кредит: 0,00 0% | Оплачено: 0,00 0% | Отгружено: 0,00 0%

Текущее состояние: **Ожидается аванс (до обеспечения)**
Текущее состояние ЭД: **Отработан**

- Вывод итоговых показателей в отдельных полях ввода

Поступление товаров и услуг (создание) - Бухгалтерия пр... (1С:Предприятие)

Поступление товаров и услуг (создание) *

Провести и закрыть | Провести | Создать на основании | Печать | Все действия | ?

Номер: от: 08.04.2011 0:00:00 | Организация: ООО "Конфетпром" ...
Вид операции: Покупка, комиссия | Подразделение: ...
Контрагент: Калинка ООО | Склад: Основной склад ...
Договор: | Зачет аванса: Автоматически ...

Товары (2) Услуги Счета расчетов Дополнительно Счет-фактура

+ Добавить | X | Подобрать товары | Все действия

N	Номенклатура	Количество	Цена	Сумма	Ставка НДС
1	Печенье "Принц"	10.000	200,00	2 000,00	18%
2	Конфеты "Ассорти"	5.000	150,00	750,00	18%

Всего: 2 750,00 НДС (в т.ч.): 419,49

Ответственный: Ромашкина Тамара Петровна | Комментарий:

При выборе способа оформления следует руководствоваться следующими критериями:

Объединение в группы	Отдельные поля ввода
Возможность копирования итоговых сумм не требуется	Необходимо предоставить возможность копирования итоговых сумм в буфер обмена
Итоговых показателей много, например, три и более	Итоговых показателей не много, например один или два

В итоговых показателях, размещаемых в нижней части формы, отражается сводная информация по содержанию документа.

В случае, если в итоговых показателях выводится информация по конкретной таблице, то рекомендуется размещать итоги сразу под ней, оформляя их в виде полей ввода.

Оформление итогов в группах

Итоговая информация - информация, необходимая для понимания документа и принятия решения о дальнейшем действии. Набор итоговой информации определяется для каждого документа в отдельном порядке.

Примеры итоговой информации: Итоговая сумма, Сумма с НДС, Сумма без НДС, Итоговая скидка (автоматическая, ручная).

Итоговая информация отражается в нижней части формы документа

- Оформляется в виде групп с рамками и с отображением заголовка. Заголовок должен кратко и однозначно описывать эту группу полей.
- Если в группу итоговой информации объединены данные с одинаковой валютой или единицей измерения, то ее название выносится в заголовок группы, а не указывается по каждому элементу
- Итоговую информацию необходимо обновлять при внесении изменений в содержание документа, а не только при записи или проведении
- Рекомендуется делать подсказку к итоговым показателям, чтобы она поясняла, откуда взялась та или иная цифра
- Для отображения положительной, нейтральной, непросроченной информации рекомендуется использовать цвет «ИтоговыеПоказателиДокументов» (22,39,121)
- Для отображения негативной, просроченной информации рекомендуется использовать цвет «ПросроченныеДанные» (178,34,34)
- В случае, если группа имеет основное итоговое значение (например, Заказано с НДС), то для него следует использовать шрифт «ОсновноеИтоговоеЗначение» (шрифт диалогов и меню, начертание «жирийный»)

Например, Сумму с НДС, Итоговый размер скидки и т.п.

Оформление итогов отдельными полями ввода

- Итоги оформляются отдельными полями ввода с установленным признаком «Только чтение»
- Выравниваются по правому краю формы.
- В заголовках полей следует выводить текст без слова «итог». Валюту в случае отражения нужно показывать после поля ввода.

Если итогов несколько, то их можно разместить в одну или несколько строк. Выбор варианта зависит от компоновки конкретной формы. Можно руководствоваться следующими соображениями:

- Поля, значения которых зависят друг от друга (например, «Всего» и «НДС (в т.ч.)»), рекомендуется всегда располагать на одной строке

Всего: руб. НДС (в т.ч.): руб.

- Поля, значения которых требуется визуально сравнивать, лучше располагать друг под другом

Выдано: руб.

Потрачено: руб.

- Если размеры формы этого не позволяют, то поля можно размещать в одну строку

Выдано: руб. Потрачено: руб.

Итоги в журналах документов

Область применения: управляемое приложение.

#std614

Итоги в журналах документов (при их наличии) рекомендуется оформлять следующим образом:

- Итоги размещаются в нижней части формы, слева
- Поля итогов оформляются как надписи
- Для выделения значений рекомендуется использовать цвет "ИтогиЖурналаЦвет" (RGB 100,100,100) и шрифт "ИтогиЖурналаШрифт" (шрифт диалогов и меню, начертание "жирийный")

Например, в журнале «Банковские выписки» в качестве итогов оформлены поля, отражающие остатки и обороты по расчетному счету за день.

Банковские выписки								
Организация: Банковский счет: Дата: Контрагент: Назначение платежа: Все действия								
000 "Конфетром"								
Дата	Поступло	Списано	Назначение платежа	Контрагент	Вид операции	Вх. номер	Вх. дата	А
01.09.2010 12:00:00	65 000.00		За услуги по договор...	Калинка ООО	Оплата от покупателя	4587	01.09.2010	
10.09.2010 00:00		550.00	Списано банком сум...	ИННОВАЦИЙ И РАЗ...	Прочее списание	12548	10.09.2010	
10.09.2010 12:00:00	200 000.00		За услуги по договор...	Транс сервис ООО	Оплата от покупателя	14568	10.09.2010	
01.10.2010 12:00:00	200 000.00		За услуги по договор...	Транс сервис ООО	Оплата от покупателя	12468	01.10.2010	
10.10.2010 12:00:00		550.00	Списано банком сум...	ИННОВАЦИЙ И РАЗ...	Прочее списание	15487	10.10.2010	
11.10.2010 00:00		100 000.00	перечислено подотчет	ИННОВАЦИЙ И РАЗ...	Перечисление подот...	145	11.10.2010	
14.10.2010 18:00:01		4 000.00	перечислено подотчет	ИННОВАЦИЙ И РАЗ...	Перечисление подот...	578	14.10.2010	
14.10.2010 18:00:02		14 000.00	Оплата по договор...	Торговый дом ООО	Оплата поставщику	4	14.10.2010	
01.11.2010 12:00:00	250 000.00		За услуги по договор...	Транс сервис ООО	Оплата от покупателя	56897	01.11.2010	
10.11.2010 12:00:00		550.00	Списано банком сум...	ИННОВАЦИЙ И РАЗ...	Прочее списание	25478	10.11.2010	
15.11.2010 0:10:01		55 000.00	перечислено подотчет	ИННОВАЦИЙ И РАЗ...	Перечисление подот...	5987	15.11.2010	
29.11.2010 12:00:00		100 000.00	перечислено подотчет	ИННОВАЦИЙ И РАЗ...	Перечисление подот...	8746	29.11.2010	
01.12.2010 12:00:00	150 000.00		За услуги по договор...	Связьинвест ООО	Оплата от покупателя	78954	01.12.2010	
06.12.2010 0:00:00	11 800.00		Оплата по счету № 1...	Тройон ООО	Оплата от покупателя	5872	06.12.2010	
06.12.2010 0:00:00		100 300.00	Оплата по договор...	Попов А.П. ИП	Оплата поставщику	3	06.12.2010	
07.12.2010 0:00:00		100 000.00	Перечислена на карт...	ИННОВАЦИЙ И РАЗ...	Оплата поставщику	1	07.12.2010	
07.12.2010 0:00:00		14 942.00	НДФЛ за ноябрь 20...	ИФНС России по г...	Прочее списание	2	07.12.2010	
21.04.2011 0:00:00	10 000.00		Возврат ошибочно п...	Торговый дом ООО	Оплата от покупателя	478	07.12.2010	
21.04.2011 0:00:00		550.00	За обслуживание р/...	ИННОВАЦИЙ И РАЗ...	Оплата поставщику	25478	07.12.2010	

Флажки

Флажки используются для отражения в формах реквизитов с типом **Булево**. Подписи к флажкам следует оформлять в соответствии с приведенными рекомендациями.

1. Подпись всегда следует располагать справа от флажков:

Правильно

Запомнить пароль

Неправильно

Запомнить пароль:

2. Если к флажку требуется сделать подсказку, то ее следует располагать либо снизу, либо справа от подписи:

Правильно

Запомнить пароль

Сохраняется при повторном входе в программу

Неправильно

Запомнить пароль: Сохраняется при повторном входе в программу

Запомнить пароль Сохраняется при повторном входе в программу

3. Первая буква подписи флажка должна быть заглавной:

Правильно

Запомнить пароль

Неправильно

запомнить пароль

4. Подписи у флажков следует делать позитивными (не содержащими отрицания)

Правильно

Использовать поиск

Неправильно

Не использовать поиск

Если при этом требуется, чтобы флажок по умолчанию был установлен, у реквизита объекта метаданных следует установить свойство **Значение заполнения** в значение **Истина**.

Подпись, содержащую отрицание, следует применять в исключительных случаях, когда это выглядит естественно или обусловлено историческими причинами.

Например, "#anchor_638">Работа с неактуальными (недействительными объектами)

5. Текст подписи должен быть понятным и кратким. Не следует делать подпись, состоящую из двух и более предложений.

6. Подпись к флажку определяет только один вариант, второй остается неявным и не сформулированным. Поэтому текст подписи следует подбирать так, чтобы у пользователей не возникало сомнений в том, каким будет второй вариант:

Правильно

Учитывать прочие активы и пассивы

Неправильно

Прочие активы и пассивы

В случае, если такую подпись к флажку подобрать не удается, лучше использовать радиокнопки, у которых указать названия явным образом, например:

Правильно

Услуга Товар

Неправильно

Услуга

7. Если на форме присутствует группа флажков, в подписях к которым используется общий текст, то его следует выносить в заголовок группы:

Правильно

Использовать:
 Дополнительные реквизиты и сведения
 Взаимодействия
 Электронные цифровые подписи
 Полнотекстовый поиск

Неправильно

Использовать дополнительные реквизиты и сведения
 Использовать взаимодействия
 Использовать электронные цифровые подписи
 Использовать полнотекстовый поиск

Команда «Подобрать»

Область применения: управляемое приложение.

#std591

- Рекомендуется использовать название команды в форме инфинитива – «Подобрать».
- Если есть уверенность в том, что пользователям удобнее и эффективнее работать с помощью «Подобрать», а не через «Добавить», то команду следует расположить первой. Если нет, то второй или в другом месте.
- Для команды «Подобрать» используйте акселератор F8.

Команда «Отмена»

Область применения: управляемое приложение.

#std592

- Команда «Отмена» должна использоваться для реальной отмены установленных настроек и приводить к закрытию формы.
- «Отмена» не должна быть кнопкой по умолчанию. Если никакие действия кроме закрытия формы недоступны, кнопкой по умолчанию должна быть кнопка «Закрыть».

Единицы измерения

Область применения: управляемое приложение.

#std593

Для полей ввода

1.1. Единицы измерения следует указывать после полей, а не в заголовке:

Хорошо	Плохо
Сумма: <input type="text" value="1000,00"/> руб.	Сумма, руб.: <input type="text" value="1000,00"/>

Хорошо	Плохо
Ставка налога: <input type="text" value="14,00"/> %	Ставка налога, %: <input type="text" value="14,00"/>

Хорошо	Плохо
Вознаграждение: <input type="text" value="30,00"/> %	Процент вознаграждения: <input type="text" value="30,00"/>

1.2. Единицы измерения следует указывать без скобок:

Хорошо	Плохо
Сумма: <input type="text" value="100,00"/> руб.	Сумма: <input type="text" value="100,00"/> (руб.)

1.3. Если к единице измерения выводится расшифровка, то ее следует располагать в скобках, после единицы измерения.

Например, для срока полезного использования:

Срок полезного использования: мес. (20 лет 5 месяцев)

Для списков

2.1. Единица измерения выводится в колонке с названием "Ед. изм".

Хорошо		Плохо	
		Количество	Ед.
		100,000	кг
		5,000	шт
Количество	Ед. изм.		
100,000	кг		
5,000	шт		
		Количество	Единица измерения
		100,000	кг
		5,000	шт

Иключение: денежные единицы измерения выводятся в колонке с названием "Валюта".

Сумма	Валюта
100,00	RUB
200,00	RUB
500,00	EUR
50 000,00	RUB
1 500,00	USD

2.2. Если единица измерения для разных значений всегда одинакова, то ее следует выводить в заголовке колонки

Хорошо	Плохо
Сумма, руб.	Сумма
20 000,00	20 000,00 руб.
5 000,00	5 000,00 руб.
100,00	100,00 руб.
1 000,00	1 000,00 руб.
Размер, кб.	Размер
41	41 кб.
75	75 кб.
34	34 кб.
28	28 кб.

2.3. Если единица измерения для разных значений отличается, то ее можно выводить как в отдельной колонке, так и в заголовке колонки с количеством (суммой). При этом единицу измерения следует выводить сразу после значения, которое она измеряет.

Хорошо			Плохо		
Наименование	Количество	Ед. изм.	Наименование	Ед. изм.	Количество
Папка-вкладыш А4	100,000	шт	Папка-вкладыш А4	шт	100,000
Клей канцелярский	5,000	кг	Клей канцелярский	кг	5,000
Наименование			Наименование		
Папка-вкладыш А4	100,000	шт	Папка-вкладыш А4, шт	100,000	
Клей канцелярский	5,000	кг	Клей канцелярский, кг	5,000	

Исключение: если одна единица измерения относится к нескольким колонкам, то ее следует располагать до этих колонок.

Например, в документе "Пересчет товаров" две колонки с количеством – по учету и по факту:

N	Номенклатура	Характеристика	Ед. изм.	По учету	Факт
1	X-1234 BOSCH Завод бытовой тех...	«характеристики не используются»	шт	1.000	1.000
2	Ассорти (конфеты)	«характеристики не используются»	упак	70.000	70.000
3	Барбарис (конфеты)	«характеристики не используются»	кг	1.000	1.000
4	Ботинки женские натуральная ко...	36, 5, натуральная кожа, Зеленый	пар	10.000	10.000
5	Ботинки женские натуральная ко...	38, 6, натуральная кожа, Зеленый	пар	10.000	10.000
6	Ботинки мужские	Размер: 42, Цвет: Черный	пар	5.000	5.000

Значения по умолчанию

Область применения: управляемое приложение.

#std594

Рекомендуется явным образом выделять среди остальных значения, используемые для подстановки в поля по умолчанию.

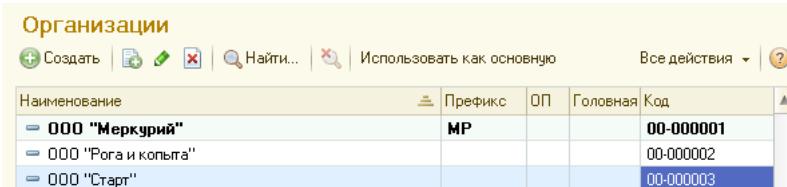
Для отражения таких значений лучше использовать термин «основной».

Выделение в списках

- Для выбора в списке основного элемента рекомендуется использовать отдельную команду, расположенную в командной панели списка
- Название для команды выбирается в зависимости от заголовка формы:
 - Использовать как основной
 - Использовать как основную
 - Использовать как основное

Например, для формы «Организация» команда будет называться «Использовать как основную»

- Для элемента, выбранного основным, используется шрифт "ОсновнойЭлементСписка" (шрифт диалогов и меню, начертание "жирный")



Выделение в форме

- Если в форме одно поле, используемое для заполнения по умолчанию, то рекомендуется после его заголовка добавлять слово «основной» (основная, основное) в круглых скобках
- Банковский счет (основной): ...
- Если в форме несколько полей, используемых для заполнения по умолчанию, их следует оформлять в виде группы с заголовком «Используются как основные». В заголовки полей слово «основной» добавлять не рекомендуется.

Используются как основные	
Банковский счет:	<input type="text"/> ... <input type="button" value="🔍"/>
Договор:	<input type="text"/> ... <input type="button" value="🔍"/>
Контактное лицо:	<input type="text"/> ... <input type="button" value="🔍"/>

В случае, если создается и заполняется форма, имеющая подчиненные формы, то при заполнении полей со значениями по умолчанию рекомендуется записывать данные автоматически, не выдавая сообщений пользователю.

Гиперссылка на счет-фактуру

Область применения: управляемое приложение.

#std595

В случае, если в форме документа есть гиперссылка на счет-фактуру, то ее рекомендуется оформлять следующим образом:

- Гиперссылка размещается в нижней части формы до группы полей «Ответственный» и «Комментарий»
- Располагается в отдельной строке с выравниванием по левому краю формы
- Перед ссылкой выводится подпись «Счет-фактура». Допускается уточнять в подписи тип счета-фактуры
Например, «Счет-фактура выставленный», «Счет-фактура полученный», «Счет-фактура на сумму вознаграждения»

- Рекомендуется размещать гиперссылку на всю ширину формы. В случае, если подвал многоколоночный и поле «счет-фактура» относится к колонке, то его следует делать соответствующим ширине колонки
- При наличии в форме нескольких гиперссылок на счета-фактуры следует размещать их друг под другом

Счет-фактура на реализованный товар: [Создать счет-фактуру](#)
 Счет-фактура на сумму вознаграждения: [Создать счет-фактуру](#)

Оформление текста гиперссылки

- Если счет-фактура не указан, рекомендуется выводить текст «Создать счет-фактуру»

Счет-фактура: [Создать счет-фактуру](#)

- Если счет-фактура указан, то выводится его номер и дата

Счет-фактура: № КФ00-000001 от 02.04.2011г.

- Если ввод счета-фактуры не требуется, рекомендуется выводить текст «Не требуется»

Счет-фактура: Не требуется

- Если для создания нового счета-фактуры требуется записать текущий объект, то рекомендуется по-возможности не выдавать сообщение типа «Документ не записан. Записать?», а автоматически записывать объект.

В ситуации, если у пользователя нет права на создание счета-фактуры, но есть право на ее чтение, рекомендуется следующее оформление:

- Вместо гиперссылки выводится надпись «Недостаточно прав для создания счета-фактуры» в угловых скобках
- Для текста надписи используется цвет «ПояснениеОтсутствующейГиперссылки» (128,128,128)

Счет-фактура: <Недостаточно прав для создания счета-фактуры>

Поле, влияющее на состав остальных полей в форме

Область применения: управляемое приложение.

#std631

В случае, если на форме есть поле, выбор значения в котором существенно влияет на состав других элементов формы, необходимо выделять его визуально.

Например, Вид операции, Вид документа и т.п.

Выделение поля требуется для привлечения внимания пользователя, чтобы показать ему, какое именно значение установлено в этом поле, т.к. это может повлиять на работу с формой.

Для выделения поля следует изменить его фон на цвет «ФонУправляющегоПоля» (255,232,179)

The screenshot shows the 'Realization of goods and services (creation)' window. At the top, there are several input fields: 'Номер:' (Number), 'от:' (from) with the value '08.07.2011 0:00:00', 'Подразделение:' (Department), 'Вид операции:' (Type of operation) which is highlighted with a red border and contains the value 'Продажа, комиссия', 'Склад:' (Warehouse) with 'Основной склад' (Main warehouse), 'Договор:' (Contract), 'Организация:' (Organization) with 'ООО "Конфетпром"', and 'Контрагент:' (Counterparty). Below these are tabs for 'Товары' (Goods), 'Услуги' (Services), 'Счета расчетов' (Accounts receivable), and 'Дополнительно' (Additional). A large table below the tabs has columns: 'N', 'Номенклатура' (Nomenclature), 'Количество' (Quantity), 'Цена' (Price), and 'Сумма' (Amount). The 'Вид операции:' field is clearly highlighted with a red border, indicating its importance for the form's logic.

Реализация товаров и услуг (создание)

Провести и закрыть | Провести | Создать на основании | Печать | Все действия | ?

Номер: от: 08.07.2011 0:00:00 | Подразделение: ... |

Вид операции: Продажа, комиссия | Склад: Основной склад |

Договор: Продажа, комиссия | Организация: ООО "Конфетпром" |

Контрагент: Оборудование |

Товары Услуги Счета расчетов Дополнительно

+ Добавить | Подбор | Все действия

N	Номенклатура	Количество	Цена	Сумма

Невыбранная картинка

Область применения: управляемое приложение.

#std635

Если в форме есть изображение (поле картинки), которое может быть добавлено пользователем, то его оформление должно сообщать пользователю о своем назначении.

Например, поле для фотографии в карточке сотрудника, поле для изображения в справочнике номенклатуры и т.п.



- Как оформлять невыбранную картинку:
Обязательно указывать текст невыбранной картинки
Например, «Добавить фото», «Добавить изображение»
- Для текста невыбранной картинки используется цвет "ТекстНевыбраннойКартинкиЦвет" (RGB 220,220,220) и шрифт "ТекстНевыбраннойКартинкиШрифт" (шрифт диалогов и меню, размер "12")

Реквизиты

Область применения: управляемое приложение.

#std649

- Рекомендуется всегда оставлять на форме стандартные поля, такие как наименование, номер и дата (для документов и задач).
- Если для реквизита предусмотрен выбор из списка, а в списке имеется только один вариант, его следует подставлять по умолчанию.

Примеры: одно значение ставки НДС (18%), один договор по контрагенту, заказ.

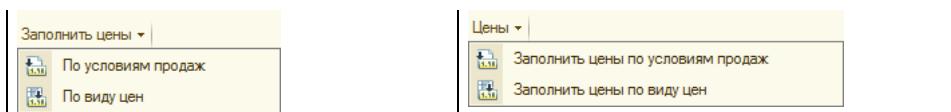
- При создании на основании необходимо заполнять все наследуемые реквизиты.

Подменю

1. При формировании командных панелей однотипные команды рекомендуется объединять в подменю.

2. Если в подменю есть команды, в заголовках которых присутствует общий текст, то его следует выносить в заголовок подменю:

Правильно	Неправильно



3. Подменю не должно содержать одну команду. В этом случае ее следует выносить в командную панель:

Правильно	Неправильно
Создать на основании Счет на оплату	Создать на основании Печать Счет на оплату

Требования к изображениям

Область применения: управляемое приложение.

#std596

- Хорошо узнаваемые метафоры, выдержаные в едином стиле.
 - Разные очертания у близких в ряду иконок, чтобы повысить их запоминаемость.
 - Соблюдаены одинаковые углы, перспектива, тени во всех используемых иконках.
 - Объекты изображены в современном виде (например, компьютер).
 - Иконки разные по начертанию и по ведущим цветам.
 - Исключите ситуации, когда различить иконки можно только по их цвету.
 - Формат - картинки PNG с 8-битным альфаканалом.
- Если есть возможность без потери качества сделать картинку с 1-битным альфа-каналом, то лучше делать именно так для уменьшения размера конфигурации.*
- Не следует использовать в конфигурациях альфа-канал у картинок, чей размер превышает 40000 точек (например, картинка 200x200 точек). Для таких картинок не поддерживается корректное отображение в веб-клиенте, который работает в веб-браузере Microsoft Internet Explorer 6.0. Это правило не относится к картинкам-коллекциям, размер элементов которых меньше указанного ограничения.
 - Если в конфигурации делается поле HTML документа, которое будет показывать какие-либо картинки, то эти картинки нужно делать в формате PNG с 1-битным, а не 8-битным альфаканалом, т.е. в формате png-8, либо в формате png-24 без альфа-канала. Для правильного отображения цветов в браузере рекомендуется удалять из заголовка файла PNG информацию о палитре и гамма-коррекции.
 - Для корректного отображения картинок конфигурации во всех поддерживаемых браузерах необходимо, чтобы используемые цвета были либо полностью прозрачными, либо полностью непрозрачными. То есть компонент их прозрачности должен быть равен либо 0% либо 100%.

Правила создания иконок командных панелей

Область применения: управляемое приложение.

#std423

Руководство по созданию иконок командных панелей. Данное руководство рассчитано на использование с продуктами **Adobe Photoshop** и **Axialis IconWorkshop**.

Палитра



821414 961717 c21600 eb3a23 ff54a f9e70 ffa5a fedab9



27c4a 294d66 24628f 4486b4 68aedc 87d1ff b4e2ff cf66ff



784c2a 8d5a32 a1714b b58864 c29d7e d5b397 ebcab5 ffe5cf



284f15 306e12 439819 56c420 83e247 afffb6 06ff94 ddffba



386178 57758a 6b2ab 93acbd a9c5d9 b9d7ed cce1f0 d9efff



716622 b5913a c7a622 f0d431 f4df45 f0dd72 faf4b3 fafde6

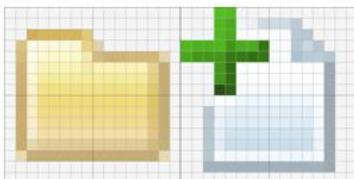
Палитра предполагает использование серого от #4D4D4D до #FFFFFF, а также цветов смешения между указанными цветами.

Проекция

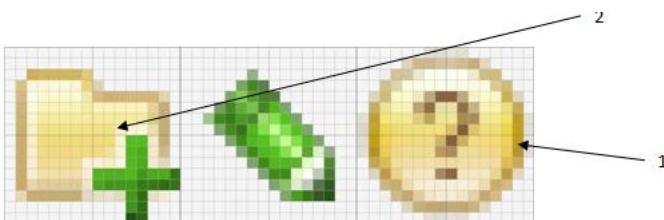
Для лучшей разборчивости элементы маленьких иконок отображаются во фронтальной проекции, без перспективы.

Размер

Иконки командной панели имеют размер 16x16. Но изображение не должно непременно занимать всё доступное пространство. Главное, чтобы иконки имели примерно одинаковый визуальный вес и чтобы их центр тяжести располагался на одной горизонтальной линии.



Элементы стиля



1. Обводка. Элементы имеют обводку толщиной 1 пиксель, с закругленными углами. Технически обводка реализуется растушевкой с альфа - каналом. Прозрачность от 10 до 50%. На углах прозрачный пиксель от 5 до 15% прозрачности.

2. Контур. Для придания объектам читаемости, от модификатора идет внешний дополнительный контур на объект, шириной 1 пиксель. Подсветка может располагаться снизу, справа - внизу или по всему периметру: на усмотрение художника. Интенсивность подсветки подбирается индивидуально.

3. Тени. Собственные и падающие тени отсутствуют.

4. Модификаторы. Имеют основной зеленый и синий цвет. Исключение составляют модификаторы «опасности».

Направление света

Источник света считается расположенным вверху. Поэтому обводка закрашивается вертикальным градиентом (обычно от 2-го до 5-го цвета из палитры). На лицевой стороне объектов допускается как вертикальный градиент, так и градиент, направленный из верхнего левого угла в нижний правый (на усмотрение художника).

Названия печатных форм учетных документов и команд по их выводу на печать

Область применения: управляемое приложение.

#std645

Рекомендации распространяются на формы первичных учетных документов, утвержденные Росстатом (Госкомстата).
Например, первичные документы по учету торговых операций, кассовых операций, материалов, кадров и т.п.

Как правило, такие учетные документы имеют в названии кодовое обозначение.

Например, «ТОРГ-12» для Товарной накладной.

Стандарт следует применять к названиям печатных форм документов, а также к заголовкам команд по их выводу на печать.

Название печатной формы и заголовок команды по выводу на печать рекомендуется делать одинаковым.

Названия необходимо оформлять единообразно, придерживаясь следующих рекомендаций:

1. Название, по возможности, следует делать кратким

Например, «Приказ о приеме (Т-1)» вместо «Приказ (распоряжение) о приеме работника на работу (Т-1)»

2. В названии можно использовать общепринятые сокращения

Например, «Акт о приеме-передаче ТМЦ»

3. В название следует включать кодовое обозначение

Например, «Товарная накладная (ТОРГ-12)», «Накладная на отпуск материалов (М-15)», «Личная карточка (Т-2)»

Кодовое обозначение следует размещать в конце названия в круглых скобках, чтобы оно не затрудняло поиск нужной команды по первым буквам.

Хорошо	Плохо
<p>Печать ▾</p> <p>Инвентаризационная опись (ИНВ-1) Сличительная ведомость (ИНВ-18) Приказ о проведении инвентаризации (ИНВ-22)</p>	<p>Печать ▾</p> <p>ИНВ-1 (Инвентаризационная опись) ИНВ-18 (Сличительная ведомость) ИНВ-22 (Приказ о проведении инвентаризации)</p>

4. Из названия должно быть понятно назначение документа. Не допускается использовать обобщенные и обезличенные словосочетания, а также указание на кодовое обозначение

Хорошо	Плохо
Расчетная ведомость (Т-51)	Унифицированная форма (Т-51) Расчетная ведомость по форме Т-51 Расчетная ведомость (ф. Т-51)

См. также: [Формирование печатных форм](#)

Тексты

Область применения: управляемое приложение.

#std598

Общие рекомендации

- Избегайте технологических терминов и аббревиатур, используйте язык пользователей.

- Следите за единобразием понятий, которые используются в программе.
- Используйте короткие предложения т.к. чтение с экрана монитора достаточно быстро утомляет.
- Если текст объемный, сначала укажите главную мысль, затем другие в соответствии с уменьшением их важности.
- Разбивайте текст на небольшие абзацы.
- Проверяйте все надписи, подписи и сообщения на правописание и грамматику.
- Пол ввода и подписи нему должны отображаться одинаковым шрифтом (начертание, размер, жирность), но не цветом.
- В интерфейсных текстах используйте только прямые кавычки. В справке допускается использовать французские кавычки «елочки».
- Буква «ё» употребляется только в тех случаях, когда возможно неправильное прочтение слова, когда надо указать правильное произношение редкого слова или предупредить речевую ошибку. Букву «ё» следует также писать в собственных именах. В остальных случаях следует писать букву «е». Подробнее см. [Грамота.Ру. Минобрнауки России разъяснил порядок применения буквы «ё» в официальных документах, правовая практика употребления буквы «ё»](#).

Названия и заголовки

- Рекомендуется избегать названий и заголовков, которые могут быть неоднозначно поняты пользователями.
- Не рекомендуется использование названий и заголовков длиной более 60 символов.
- Термином «родитель» обозначайте родственные отношения между физическими лицами, но не для обозначения иерархии объектов информационной базы. Вместо этого можно использовать прикладные термины, например:
 - группа товаров
 - головное подразделение
 - входит в группу
 - входит в категорию
 - находится в группе и т.д.
- Если в конфигурации принято выводить количество элементов (например, "Товары(10)") в заголовках страниц или групп, то этот прием должен применяться везде, во всех объектах конфигурации.

Когда использовать многоточие

- Используйте многоточие, если название команды или ссылки отражает действие («Изменить форму», «Распределить задачи» – в форме инфинитива или отглагольного существительного), а при нажатии осуществляется переход к дополнительной форме (месту) для выполнения этих действий.
 - Многоточие должно прымывать к последнему слову без пробела, например «Печать...», а не «Печать ...».
 - Если название команды описывает процесс, который будет запущен, то многоточие не нужно.
- [«Изменение формы»](#), [«Распределение задач»](#) и пр.

Концевые пробелы в представлениях объектов и заголовках элементов управляемых форм

- Не допускается использование концевых пробельных символов (пробел, неразрывный пробел, табуляция) в текстах заголовков элементов управляемых форм и представлениях объектов конфигурации (представление и расширенное представление объекта, списка, записи).

Представления объектов и заголовки элементов управляемых форм применяются при отрисовке управляемого интерфейса и управляемых форм и концевые (как левые, так и правые) пробельные символы могут искажать отрисовку форм, панели действия, панели навигации, панели разделов.

Ограничения по использованию одинаковых текстов на элементах управления в форме

- Рекомендуется воздерживаться от размещения в форме элементов управления с одинаковым текстом. Например, два подменю "Все действия" в табличной части и в самой форме. Совпадение текстов элементов одного типа на форме создает путаницу в действиях пользователя, затрудняет чтение и написание справки, а также может привести к неопределенности при объяснении пользователем своих действий в форме другим пользователям или специалистам.

См. также

- [Имя, синоним, комментарий](#)
- [Подсказка и проверка заполнения](#)

Шрифт и цвет

Область применения: управляемое приложение.

#std660

- Не рекомендуется использование абсолютных шрифтов и цветов. Вместо этого следует создать соответствующие элементы стиля.
- Не рекомендуется использовать подчеркивание для элементов формы, не являющихся гиперссылкой.
- Не рекомендуется использование прописных букв в текстах (кроме первой буквы предложений).

Горячие клавиши

Область применения: управляемое приложение.

#std665

- Рекомендуется установить сочетание клавиш для наиболее частотных команд.
- Рекомендуется установить сочетание клавиш для вызова часто используемых неконтекстных форм.

[Например, форма полнотекстового поиска, форма ежедневного отчета о рабочем времени.](#)

Элементы стиля

Область применения: управляемое приложение.

#std667

Для каждого элемента управления оформление по умолчанию задается платформой. Умолчаний следует придерживаться в большинстве случаев — это позволяет обеспечить единобразное оформление всех форм.

В некоторых ситуациях возникает потребность визуально выделить конкретный элемент управления среди других, изменив его оформление по умолчанию.

Для изменения оформления следует использовать элементы стиля, а не задавать конкретные значения непосредственно в элементах управления. Это требуется для того, чтобы аналогичные элементы управления выглядели одинаково во всех формах, где они встречаются.

Виды элементов стиля:

- Цвет (задается значение RGB)
- Шрифт (задаются вид, размер и начертание)
- Рамка (задаются тип и ширина границ)

1. Элементы стиля нужно использовать всегда, когда требуется изменить оформление (Цвет, Шрифт, Рамку), установленные по умолчанию.

Например, информационные надписи среди других надписей можно выделять с помощью цвета. Цвет таких надписей следует задавать в виде элемента стиля "ИнформационнаяНадпись", а не в виде значения RGB или выбора цвета web/windows:

Хорошо	Плохо
ЦветТекста <input type="color"/> стиль: ИнформационнаяНадпись <input type="button" value="..."/> <input type="button" value="X"/>	ЦветТекста <input type="color"/> 70, 130, 180 <input type="button" value="..."/> <input type="button" value="X"/> ЦветТекста <input type="color"/> Синий со стальным оттенком (SteelE <input type="button" value="..."/> <input type="button" value="X"/>

2. Не следует использовать элементы стиля для того, чтобы подменить оформление, которое используется по умолчанию в платформе.

Например, для гиперссылок нужно использовать цвет, предусмотренный в платформе, а не создавать для него свой элемент стиля с точно таким же цветом:

Хорошо	Плохо
ЦветТекста <input type="color"/> Авто <input type="button" value="..."/> <input type="button" value="X"/>	ЦветТекста <input type="color"/> стиль: Гиперссылка <input type="button" value="..."/> <input type="button" value="X"/>

3. Каждый элемент стиля следует создавать для применения в конкретной ситуации. Если такой же цвет или шрифт нужно использовать в другой ситуации, то для нее нужно создать отдельный элемент стиля.

Например, цвет элемента стиля "ФонУправляющегоПоля" следует применять только для фона полей, которые влияют на видимость других полей в форме. Если такой же цвет фона предполагается использовать для поля с другим назначением, то для него нужно создать отдельный элемент стиля.

4. Название для элемента стиля следует подбирать таким образом, чтобы в нем отразить назначение элемента стиля.

Например:

Название элемента стиля	Назначение
ФонУправляющегоПоля	Фон поля, которое управляет видимостью других полей
ТекстНевыбраннойКартинки	Текст, который будет отражаться на картинке, пока она не выбрана

5. Для нескольких элементов стиля, имеющих одинаковое название, но разный вид, рекомендуется включать вид (слова "Цвет", "Шрифт", "Рамка") в название:

Например: "ТекстНевыбраннойКартинкиЦвет" и "ТекстНевыбраннойКартинкиШрифт".

При этом вид элемента стиля (Цвет, Шрифт, Рамка) следует указывать после его названия. Это требуется для того, чтобы можно было по первым буквам найти нужный элемент стиля в списке.

Например:

Хорошо	Плохо
ПросорченныеДанныеЦвет	ЦветПросорченныхДанных
ПросорченныеДанныеШрифт	ШрифтПросорченныхДанных

В названии элемента стиля следует указывать только тот вид (Цвет, Шрифт, Рамка), который используется фактически.

Например, для элемента стиля вида "Цвет" не следует включать в название слово "Шрифт":

Хорошо	Плохо
ВажнаяНадпись ▼Оформление: ЦветТекста <input type="color"/> стиль: ВажнаяНадпись <input type="button" value="..."/> <input type="button" value="X"/> Шрифт <input type="color"/> Авто <input type="button" value="..."/> <input type="button" value="X"/>	ШрифтВажнойНадписи ▼Оформление: ЦветТекста <input type="color"/> стиль: ШрифтВажнойНадписи <input type="button" value="..."/> <input type="button" value="X"/> Шрифт <input type="color"/> Авто <input type="button" value="..."/> <input type="button" value="X"/>

Элементы стиля с видом "Цвет"

Элемент стиля	Значение (RGB)	В каком стандарте используется
ПросорченныеДанные	178,34,34	Акцентирование внимания на просорченных или критичных состояниях Итоги в документах
ПояснениеОтсутствующейГиперссылки	128,128,128	Гиперссылка на счет-фактуру
ТекстЗапрещеннойЯчейки	192,192,192	Пояснение невозможности заполнения ячеек в табличных частях
ИтоговыеПоказателиДокументов	22,39,121	Итоги в документах
ИтогиЖурналаЦвет	100,100,100	Итоги в журналах документов
ФонУправляющего поля	255,232,179	Поле, влияющее на состав остальных полей в форме
ТекстНевыбраннойКартинкиЦвет	220,220,220	Невыбранная картинка
НегативноеСобытие	178,34,34	Отчеты вида "таблица", "список"
ПозитивноеСобытие	0,128,0	Отчеты вида "таблица", "список"

НеактуальнаяИнформация	255,200,200	Отчеты вида "таблица", "список"
Диаграмма	70,130,180	Отчеты вида "диаграмма"
Прогноз	199,21,133	Отчеты вида "диаграмма"
ПоясняющийТекст	128,122,89	Рабочее место

Элементы стиля с видом "Шрифт"

Элемент стиля	Значение (шрифт, размер, начертание)	В каком стандарте используется
ОсновнойЭлементСписка	Шрифт диалогов и меню, начертание: Полужирный	Значения по умолчанию
ОсновноеИтоговоеЗначение	Шрифт диалогов и меню, начертание: Полужирный	Итоги в документах
ИтогиЖурналаШрифт	Шрифт диалогов и меню, начертание: Полужирный	Итоги в журналах документов
ТекстНевыбраннойКартинкиШрифт	Шрифт диалогов и меню, размер: 12	Невыбранная картинка

Общие правила построения интерфейсов

Область применения: обычное приложение.

#std500

Обычное приложение

1. Для каждой укрупненной группы пользователей (рабочего места) в конфигурации рекомендуется определять отдельный интерфейс: главное меню, набор и состав панелей инструментов.
- 2.1. Главное меню обеспечивает доступ ко всем формам, которые требуются пользователю для работы или сервисных функций. В подменю с большим числом элементов, группы элементов должны быть ограничены разделителями. Максимальное число элементов в одной группе не более 7.
- 2.2. При любом положении выбора действия из главного меню, его размер должен быть таким, чтобы полностью умещаться на экране при минимальном разрешении, из расчета на которое разработана конфигурация.
- 2.3. По умолчанию пункты меню для доступа к справочникам, документам, планам видов характеристик, планам счетов, планам видов расчетов и регистрам вызывают формы списков этих объектов.
- 3.1. Интерфейс следует проектировать таким образом, чтобы группе пользователей, с одной стороны, был доступен необходимый набор действий, а с другой, не предоставлялся доступ к действиям на которые нет прав. Вызовы наиболее часто выполняемых пользователем действий в интерфейсе лучше располагать так, чтобы они были наиболее доступны, и наоборот.
- 3.2. Желательно однотипные блоки меню и панели инструментов в разных интерфейсах делать похожим образом.
4. В каждой конфигурации обязательно должны быть интерфейсы **Общий** и **Полный**.

См. также

- [Общие интерфейсы](#)
- [Интерфейс "Полный"](#)

Общие интерфейсы

Область применения: обычное приложение.

#std501

Обычное приложение

1. Для отображения общих для всех переключаемых прикладных интерфейсов пунктов меню и панелей инструментов в конфигурации создается интерфейс **Общий**. У него снимается признак **Переключаемый**, а в качестве подсистемы указывается вся конфигурация.
2. На верхнем уровне главного меню должны обязательно располагаться в крайней левой позиции подменю **Файл** и в крайней правой позиции набор подменю **Сервис, Окна, Справка**.
3. В меню **Сервис** может присутствовать подменю **Переключить интерфейс** для переключения текущего интерфейса на другой. Подменю должно содержать список всех переключаемых интерфейсов конфигурации.
4. В интерфейсе **Общий** рекомендуется отключать подменю **Операции**, кроме случаев, когда это подменю используется во всех переключаемых интерфейсах конфигурации.

5. В конфигурации может быть создано несколько непереключаемых интерфейсов для того, чтобы реализовать зависимость состава главного меню и панелей инструментов от роли пользователя. Для этого состав интерфейсов конфигурации дополняется интерфейсами, названия которых начинаются со слова **Общий...**. Для таких интерфейсов снимается признак **Переключаемый** и право на их использование дается тем или иным ролям.

Интерфейс "Полный"

Область применения: обычное приложение.

#std502

Обычное приложение

1. Интерфейс **Полный** организован в соответствии с функциональными возможностями конфигурации и предоставляет доступ ко всем возможным действиям.
2. На верхнем уровне главного меню группировка действий производится по крупным функциональным группам, соответствующим одной или нескольким подсистемам. Например: **Зарплата, Бухгалтерия, Производство**.
3. На верхнем уровне обязательно присутствие подменю **Операции**, если оно не включено в состав интерфейса **Общий**.
4. Запрещается группировать подменю по существующим в конфигурации прикладным объектам, например, **Справочники, Документы, Отчеты** и т. д.

СТИЛИ

Область применения: обычное приложение.

#std524

Обычное приложение

1. Для оформления конфигурации используется стиль **Основной**. Он устанавливается в целом для конфигурации (в свойстве **Основной стиль** объекта метаданных конфигурации). Для форм конфигурации устанавливается стиль **Авто**.

2. Кроме элементов стиля, определенных платформой **1С:Предприятие**, для оформления используются 5 элементов стиля, определенных в конфигурации:

- Цвет **ТекстИнформационнойНадписи** - цвет текста информационных надписей;
- Цвет **ТекстПредупреждающейНадписи** - цвет текста предупреждающих надписей;
- Шрифт **ШрифтВажнойНадписи** - шрифт текста важной надписи;
- Цвет **ТекстВторостепеннойНадписи** - цвет текста второстепенных надписей;
- Цвет **ЦветГиперссылки** - цвет текста гиперссылки.

Имя элемента управления

Область применения: обычное приложение.

#std503

Обычное приложение

1. Для элементов управления с непустыми данными имя элемента управления должно совпадать с данными.

2. Для элементов управления с пустыми данными, но непустыми данными флагка имя элемента управления должно совпадать с данными флагка.

3. Если данные (данные флагка) указаны как путь через точку, тогда имя элемента управления образуется из имени данных (данных флагка) исключая точки.

4. В случаях когда нет или не указаны данные и данные флагка, имя элемента формы составляется из названия типа элемента управления и краткой расшифровки, поясняющей назначение элемента, например, **ПолеВыбораУсловияОтбора**.

5. Если в форме в соответствии с правилом для нескольких элементов управления образуются одинаковые имена, один из элементов управления называется по правилу, а имена остальных образуются из имени по правилу с добавлением цифры, последовательно, начиная с "1". Например, если поле ввода с данными **Номер** и подпись к нему отображаются на нескольких страницах панели в форме, то на разных страницах их нужно называть так:

- Номер, НадписьНомер;
- Номер1, НадписьНомер1;
- Номер2, НадписьНомер2;
- Номер3, НадписьНомер3;
- и т.д.

Изменения размера колонки табличного поля

Область применения: обычное приложение.

#std504

Обычное приложение

Для колонок табличных полей свойство **ИзменениеРазмераКолонки** устанавливается по следующим правилам.

Колонке разрешается изменение размера, когда длина представления данных колонки заранее не известна или когда длина данных настолько большая, что ширину колонки по умолчанию устанавливают заведомо меньше требуемой. Например, разрешается изменять размер колонки, в которой отображаются ссылочные значения, строки неограниченной длины и т.п.

Колонке не разрешается изменять размер, когда длина представления данных известна и может быть установлена при разработке формы. Например, не разрешается изменять размер колонок отображающих даты, числа, булевые значения. Исключением являются колонки, в которых отображается иерархия. Для таких колонок изменение размера запрещать нельзя.

Ограничения по использованию одинаковых текстов на элементах управления в форме

Область применения: обычное приложение.

#std526

Обычное приложение

Рекомендуется воздерживаться от размещения в форме элементов управления с одинаковым текстом. Например, два подменю **Действие**: в табличной части и в самой форме. Совпадение текстов элементов одного типа на форме создает путаницу в действиях пользователя, затрудняет чтение и написание справки, а также может привести к неопределенности при объяснении пользователем своих действий в форме другим пользователям или специалистам.

В качестве примера приведем форму по обработке набора персонала, где изначально размещено три подменю **Действия** - для формы и для каждого из табличных полей.

Набор персонала

Действия | Отчеты | Кадровое планирование... | Вакансии... | Источники информации... | Состояния заявок кандидатов... | ? |

Не разобранные письма (0)

Скрыть не разобранные письма

Действия | Принять почту | Упорядочить письма по... | Все письма...

Тема	Отправитель
Нет не разобранных писем	

Активные заявки кандидатов (2)

Действия | Упорядочить заявки по... | Все заявки...

Описание заявки		
<p>Заявка рассматривается с 9 сентября. Текущее состояние заявки: Предварительно рассмотрена. Источник, из которого кандидат узнал о вакансии: Газета "Работа для Вас". Кандидат рассматривается на вакансию: Художник, Рекламная служба. Заявка отнесена к группе: Специалисты</p>		
Переписка по заявке		
0	От кого	Кому
Дата отправления		
Нет писем по заявке кандидата		

Из варианта этой же формы, где осталось подменю **Действие** только для формы в целом.

Набор персонала

Действия | Отчеты | Кадровое планирование... | Вакансии... | Источники информации... | Состояния заявок кандидатов... | ? |

Не разобранные письма (0)

Скрыть не разобранные письма

Письма | Принять почту | Упорядочить письма по... | Все письма...

Тема	Отправитель
Нет не разобранных писем	

Активные заявки кандидатов (2)

Заявки | Упорядочить заявки по... | Все заявки...

Описание заявки		
<p>Заявка рассматривается с 9 сентября. Текущее состояние заявки: Предварительно рассмотрена. Источник, из которого кандидат узнал о вакансии: Газета "Работа для Вас". Кандидат рассматривается на вакансию: Художник, Рекламная служба. Заявка отнесена к группе: Специалисты</p>		
Переписка по заявке		
0	От кого	Кому
Дата отправления		
Нет писем по заявке кандидата		

Размеры формы

Область применения: обычное приложение.

#std505

Обычное приложение

Формы объектов метаданных типовых конфигураций разрабатываются в расчете на следующие условия применения:

- разрешение экрана 1280x768;
- окно 1С:Предприятия растянуто на весь экран;
- включено главное меню;
- включена одна строка панелей инструментов;
- включена панель окон;
- включена панель состояния.

Исходя из этого максимальными размерами формы должны быть:

- ширина - 1256 точек;
- высота – 580 точек.

Рекомендуется устанавливать оптимальные для работы пользователя размеры формы не превышая при этом приведенные размеры. Для форм, где присутствуют панели с закладками, ширину формы рекомендуется делать такой, чтобы заголовки всех закладок без сокращений были одновременно видны в форме.

Подсказки

Область применения: обычное приложение.

#std506

Обычное приложение

С целью пояснить назначение элемента управления ему назначается подсказка. Подсказка должна быть лаконичной, длина текста подсказки не должна превышать 8 слов. Заполнение подсказки обязательно.

Использование флагов "Автовыбор незаполненного" и "Автоотметка незаполненного"

Область применения: обычное приложение.

#std507

Обычное приложение

Для полей ввода, необходимо устанавливать флаг **Автовыбор незаполненного**, если установлен флаг **Автоотметка незаполненного**.
При этом свойство **Режим выбора незаполненного** устанавливается в значение **При нажатии Enter**.

Это относится в равной степени как к элементам управления, которые располагаются непосредственно в форме, так и к тем, которые принадлежат колонке табличного поля.

Иключение составляют поля ввода, в которых вводится число или дата. Для таких полей использование флага **Автовыбор незаполненного** определяется логикой работы.

Использование гиперссылок в диалогах форм

Область применения: обычное приложение.

#std508

Обычное приложение

В случае размещения команд вне командных панелей на форме, при выборе элемента управления нужно руководствоваться следующими правилами:

Используются кнопки:

- если команда изменяет или обрабатывает данные формы, особенно, если это необратимые изменения (например, очистка табличных частей);
- если это команда, которая меняет данные информационной базы, особенно если это необратимые изменения;
- если это команда выбора, подбора, заполнения.

Используются гиперссылки:

- если команда вызывает переход к другой немодальной форме или другой странице формы.

Привязки

Область применения: обычное приложение.

#std509

Обычное приложение

Необходимо минимизировать ручную установку привязок.

При горизонтальном растягивании формы

Не меняют размеров:

- надписи, за исключением "информационных надписей" и надписей с гиперссылкой,
- флаги,
- переключатели,
- кнопки,
- поля ввода в том случае, если в них водится значение заранее известной длины и это значение полностью видно в поле ввода, например: код элемента справочника, и т.д.

Изменяют размер:

- поля ввода, кроме случаев, когда они не изменяют размер,
- "информационные надписи",
- надписи с гиперссылкой,
- табличные поля,
- поля списка,
- полосы регулирования,
- командные панели,
- панели,
- рамки групп,
- индикаторы,
- поля HTML документов,
- диаграммы,
- диаграммы Ганта,
- дендрограммы,
- сводные диаграммы,
- поля картинок,
- поля табличных документов,
- поля текстовых документов.

Размер изменяется по правилам:

- если в "строке" диалога расположен один элемент, он растягивается вместе с формой;
- если в "строке" диалога расположены несколько элементов - они растягиваются пропорционально размеру формы;
- при организации информации в форме в две колонки (левая и правая) правая граница левой колонки изменяет размер пропорционально размеру формы.

При вертикальном растягивании

Не меняют размеров:

- надписи,
- поля ввода, кроме полей с многострочным вводом,
- флажки,
- переключатели,
- кнопки.

Изменяют размер:

- поля ввода с многострочным вводом,
- табличные поля,
- поля списка,
- панели,
- рамки групп,
- поля HTML документов,
- диаграммы,
- диаграммы Ганта,
- дендограммы,
- сводные диаграммы,
- поля картинок,
- поля табличных документов,
- поля текстовых документов.

Размер изменяется по правилам:

- если в "столбце" диалога расположен один элемента - он растягивается вместе с формой;
- если в "столбце" диалога расположен не один элемента - они растягиваются вместе с формой пропорционально.

В случае использования в форме вертикального разделителя, все элементы формы слева от разделителя привязываются к левой границе разделителя, как к правому краю формы, а все элементы справа от разделителя привязываются к правой границе разделителя, как к левому краю формы. По таким же правилам выполняются привязки внутри панели.

В случае использования в форме горизонтального разделителя, все элементы формы сверху от разделителя привязываются к верхней границе разделителя, как к нижнему краю формы, а все элементы снизу от разделителя привязываются к нижней границе разделителя, как к верхнему краю формы. По таким же правилам выполняются привязки внутри панели.

Отступы

Область применения: обычное приложение.

#std510

Обычное приложение

При размещении элементов управления рекомендуется использовать отступы от края формы или панели. Величину отступа рекомендуется делать такой, которая устанавливается по умолчанию при создании новой формы или новой панели.

Для формы рекомендуется устанавливать следующие отступы:

- сверху 8 или 33, если размещена верхняя командная панель;
- снизу 8 или 33, если размещена нижняя командная панель;
- справа 8;
- слева 8.

Для панели рекомендуется устанавливать следующие отступы:

- сверху 6;
- снизу 6;
- справа 6;
- слева 6.

Использование закладок

Область применения: обычное приложение.

#std511

Обычное приложение

Запрещается делать несколько страниц у самой формы.

У панелей рекомендуется делать не более 5 одновременно видимых закладок. Расположение закладок – сверху.

Допускается большее количество закладок многостраничной панели и/или расположение закладок слева:

- для специализированных форм настроек параметров;
- для форм, в которых страницы панели генерируются программно.

Порядок обхода элементов диалога

Область применения: обычное приложение.

#std512

Обычное приложение

Для всех форм устанавливается свойство **Автопорядок обхода**, кроме исключительных случаев, когда автоматически поддерживаемый порядок обхода нарушает логику работы с формой.

Обход осуществляется сверху вниз, слева направо.

Для начала обхода с произвольного элемента формы, этому элементу устанавливается признак **Активизировать по умолчанию**.

При наличии групп элементов: переключателей; флажков; групп элементов формы, выделенных рамкой или надписью, такие группы элементов в порядке обхода необходимо считать одним элементом. После первого элемента группы сначала выполняется обход остальных элементов группы, и только после этого - переход к следующему за группой по порядку элементу формы.

Размещение кнопки вызова справки в формах

Область применения: обычное приложение.

#std513

Обычное приложение

Кнопку вызова справки в формах необходимо размещать в верхней командной панели формы. Если у формы отсутствует верхняя командная панель, то кнопку размещают в любой другой командной панели, которая является для данной формы основной или наиболее часто используемой.

Допускается размещение кнопки вызова помощи, не в командной панели, а отдельно в форме.

При размещении кнопки в командной панели с выравниванием кнопок **Лево**, кнопка вызова справки должна размещаться, по возможности, в крайнем правом положении.

При других вариантах выравнивания кнопок или использовании кнопки вызова справки вне командной панели, необходимо принимать решение исходя из наилучшей эргономики формы.

Отображение единственного табличного поля в форме

Область применения: обычное приложение.

#std514

Обычное приложение

Для единственного табличного поля в форме необходимо дополнительно размещать рамку группы с названием данных, которые содержит это табличное поле. Рамка группы располагается над командной панелью табличного поля без отступа. Для рамки группы устанавливается рамка **Подчеркивание**.

Текст рамки группы должен иметь осмысленное, понятное содержание. Например, для табличного поля с табличной частью **ОС**, устанавливается текст **Основные средства**.

Разделители

Область применения: обычное приложение.

#std515

Обычное приложение

При построении форм используются вертикальные и горизонтальные разделители. Вертикальным разделителям устанавливается ширина 6, горизонтальной привязку рекомендуется устанавливать в **Непривязано**. Горизонтальным разделителям устанавливается высота 6, вертикальную привязку рекомендуется устанавливать в **Непривязано**.

Рамку рекомендуется устанавливать в **НетРамки**. В сложных случаях, когда пользователю может быть тяжело догадаться о наличие разделителя, допускается устанавливать рамку **Одинарная**.

Наличие разделителя должно интуитивно угадываться при работе с формой, поэтому ближайшие от разделителя по направлениям его передвижения элементы управления размещаются вплотную к нему, таким образом визуально предполагая его расположение.

Кнопки

Область применения: обычное приложение.

#std516

Обычное приложение

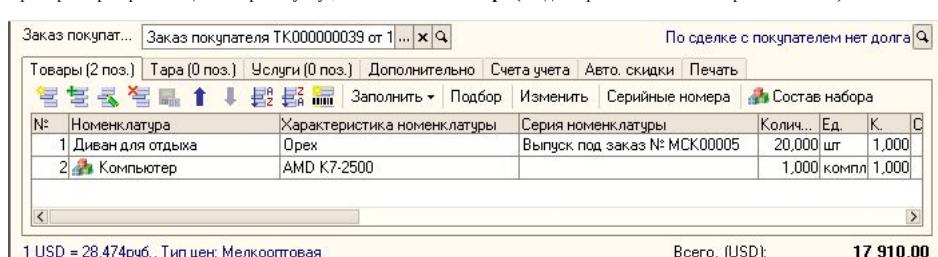
Кнопки размещаются вне командных панелей. Минимальный размер кнопок: 60x19. Ширина кнопки может увеличиваться в зависимости от длины строки заголовка кнопки. Надпись на кнопке должна занимать одну строку.

Оформление кнопок с часто употребимыми действиями (частотных кнопок)

Для оформления кнопок с часто употребимыми действиями (частотных кнопок) рекомендуется использовать текст совместно с картинкой.

Иконка, размещенная на кнопке, также может использоваться в строке табличной части для дополнительной идентификации типа данных, к которому применима данная команда.

Пример: Форма реализации товаров и услуг, кнопка **Состав набора** ("Редактировать состава набора комплекта").



Картинки

Область применения: обычное приложение.

#std517

Обычное приложение

При разработке форм типовых конфигураций используются только стандартные картинки и картинки из конфигурации. Картинки из файла (файла-коллекции) не используются. В качестве картинок действий и объектов, для которых предусмотрены стандартные картинки, допускается использовать только соответствующие стандартные картинки.

Для картинок кнопок выбора в полях ввода используются картинки размером 9x15. Для картинок кнопок используются картинки размером 16x16 или стандартные картинки размером не более 16x16. Свойство кнопки **РазмерКартинки** устанавливается в значение **РеальныйРазмер**. Для картинок кнопок командных панелей (действий и подменю) используются картинки размером 16x16.

Программное управление видимостью страниц

Область применения: обычное приложение.

#std518

Обычное приложение

При программном управлении видимостью страниц нужно делать так, чтобы в любой момент времени хотя бы одна страница была видимой: сначала нужно включать все видимые страницы, а только затем выключать все невидимые.

Игнорирование этого правила может привести к тому, что перестанут срабатывать привязки элементов управления.

Программное управление формой

Область применения: обычное приложение.

#std519

Обычное приложение

Обращение к процедурам, функциям, реквизитам, свойствам и методам, доступных для формы, из модуля этой формы происходит напрямую, без использования объекта ЭтаФорма, кроме случаев, когда такое обращение не может быть выполнено иначе.

В разделе инициализации модуля формы запрещается открывать другие формы или диалоги (например методами Вопрос, Предупреждение и т. д.).

Программное управление формой из других модулей производится через присвоение её реквизитам (свойствам) значений и через вызов её методов или экспортных процедур (функций). Не допускается делать предположений о свойствах реквизитов формы. Не допускается обращение к элементам формы не из модуля этой формы: ни непосредственно, ни при помощи перебора коллекции ЭлементыФормы, ни каким-либо другим способом. Например, вполне корректно предполагать, что у формы элемента справочника есть свойство ПараметрОснование, однако, предположение о наличии у ПараметрОснование свойства Дата, уже недопустимо.

Ограничение выполнения действий, доступных только при определенных условиях

Область применения: обычное приложение.

#std520

Обычное приложение

Если действия, выполняемые при нажатии кнопки в форме (командной панели) доступны пользователю только при наступлении определенных условий, то нажатие на кнопку должно быть доступно в любом случае.

Проверка, может ли пользователь выполнять данное действие, производится уже непосредственно в обработчике события нажатия на кнопку. Если действие доступно пользователю, оно выполняется, если нет, то пользователю коротко и понятно объясняется причина невозможности выполнения такого действия.

Например, если нельзя записывать элемент номенклатуры без заполненной базовой единицы измерения, то кнопка Записать остается доступной всегда, а при нажатии на неё, если не заполнена базовая единица измерения, может быть выдано сообщение:

Перед записью в элементе справочника "Номенклатура" необходимо заполнить реквизит "базовая единица"!

Поведение специализированных форм

Область применения: обычное приложение.

#std521

Обычное приложение

Если правильная и стабильная работа формы подразумевает некоторые особые условия, то рекомендуется завершать ее работу с выдачей соответствующего диагностического сообщения при попытке открыть ее в условиях, отличных от тех, на работу которых она рассчитана.

Например, работа формы сводного отчета о контрагенте рассчитана на вызов функции формы контрагента для получения исходных данных. При попытке открыть ее не из формы контрагента, она должна выдавать сообщение, что данная форма вызывается при редактировании контрагента в форме по кнопке Вывести сводный отчет.

Обращение к данным информационной базы в обработчиках часто вызываемых событий

Область применения: обычное приложение.

#std522

Обычное приложение

Следует минимизировать обращение к данным информационной базы в обработчиках событий, приведенных ниже, поскольку это может существенно замедлить интерактивную работу.

События формы:

- ОбновлениеОтображения.

События табличного поля:

- ПриВыходеСтроки;
- ПриАктивацииСтроки.

В качестве средств минимизации в зависимости от ситуации могут быть:

- использование переменных модуля формы для кэширования данных;
- перенос обработки данных в обработчики других событий;
- для таблиц значений - получение необходимых данных на этапе заполнения;
- любые другие методы.

Обращение к свойству "ТекущаяСтрока" табличного поля

Область применения: обычное приложение.

#std523

Обычное приложение

Запрещается использовать свойство ТекущаяСтрока для получения значений полей строки табличного поля. Обращение к данным значениям должно выполняться через ТекущиеДанные или ДанныеСтроки.

Правильно:

ИнформационнаяНадписьАдрес = "Адрес: " + Элемент.ТекущиеДанные.Адрес;

Неправильно:

Информационная надпись Адрес = "Адрес: " + Элемент.ТекущаяСтрока.Адрес;

При этом следует учитывать, что для динамических списков возможность обращения к значениям полей с помощью свойства Текущие Данные зависит от видимости соответствующих колонок в списке. Поэтому, необходимо явно добавлять данные колонки в источник данных табличного поля перед открытием формы, например:

Справочник Список.Колонки.Добавить("Адрес");

Данное правило не относится к полям, необходимым для функционирования динамических списков и расширений табличного поля (т.н. системные поля, например: Пометка Удаления, ЭтоГруппа, Дата и т.д.). Такие поля являются всегда доступными и не удаляются табличным полем из коллекции колонок динамического списка при изменении видимости или удалении колонок табличного поля.

Использование пояснений в полях ввода и выбора

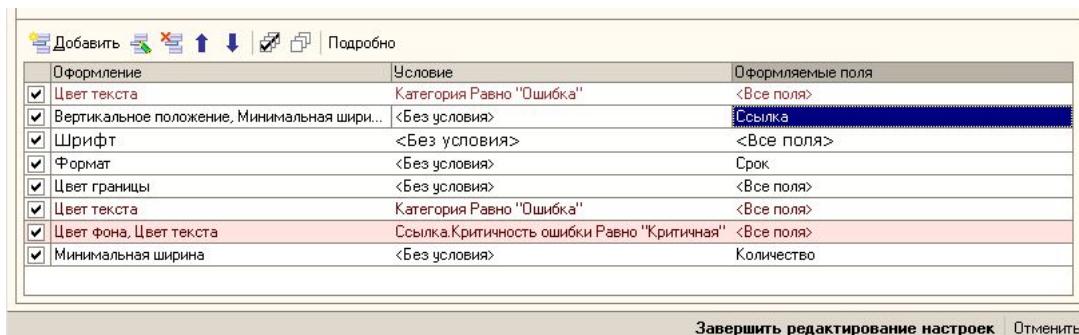
Область применения: обычное приложение.

#std527

Обычное приложение

Рекомендуется в полях ввода или выбора, в случае, если они еще не заполнены, показывать пользователю значение по умолчанию или трактовку пустого значения в этом поле.

Пример:



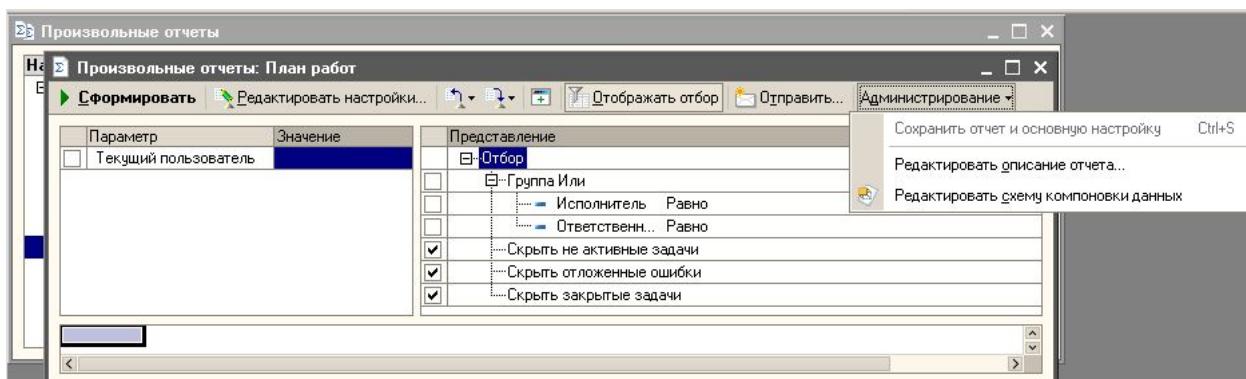
Особенности размещения в командных панелях пунктов меню, не предназначенные для решения основных задач

Область применения: обычное приложение.

#std528

Обычное приложение

Рекомендуется пункты меню, не предназначенные для решения основных задач, а исполняющих команды, предназначенные для специализированных целей или особой группы пользователей, помещать в подменю.



Вопрос при закрытии программы

Область применения: обычное приложение.

#std417

В конфигурации должна быть предусмотрена возможность исключения случайных выходов из программы "1С:Предприятие", например, если пользователь по ошибке нажал кнопку закрытия главного окна программы.

При выходе из программы необходимо задавать пользователю вопрос: "Завершить работу с программой?". Если пользователь подтверждает выход из программы - программа закрывается. Если отказывается от выхода - он продолжает работу с программой.

При этом необходимо предусмотреть возможность отключать вывод вопроса в персональных настройках пользователя.