Глава 3. Настройка и использование подсистем при разработке конфигурации

В целях обеспечения обратной совместимости во всех подсистемах библиотеки предусмотрен программный интерфейс. К нему относятся объекты метаданных библиотеки, которые предназначены для использования в прикладном коде:

- имена и состав параметров экспортных процедур и функций общих модулей, модулей объектов, менеджеров, наборов записей и т. п., которые размещены в области `ПрограммныйИнтерфейс`;
- имена и состав параметров всех экспортных процедур и функций переопределяемых общих модулей;
- имена объектов метаданных (включая их реквизиты, табличные части и пр.), к которым допускается непосредственное обращение из прикладного кода или из запросов.

В прикладном решении рекомендуется использовать программный интерфейс библиотеки, с тем чтобы существенно минимизировать затраты при переходе на новые версии библиотеки, так как при этом не требуется пересматривать код и адаптировать объекты метаданных прикладного решения под новые требования и возможности библиотеки. Прикладное решение может «уверенно» использовать старые возможности библиотеки, не «торопясь» переходить на новые.

В исключительных случаях, когда обратная совместимость не поддерживается, в файле **UpdateSSL** приведена дополнительная инструкция для разработчиков по адаптации своих конфигураций к новому программному интерфейсу библиотеки. При этом не документируются изменения в служебных процедурах и функциях библиотеки (даже если они являются экспортными), которые не относятся к программному интерфейсу. При непосредственном вызове их из прикладного кода следует учитывать, что они могут быть изменены, перемещены или удалены в следующей версии библиотеки, поскольку являются ее внутренней реализацией.

Адресный классификатор

Подсистема **Адресный классификатор** предназначена для загрузки, хранения и предоставления другим подсистемам конфигурации адресных сведений Федеральной информационной адресной системы ФНС (ФИАС). При наличии постоянного подключения к Интернету адресные сведения не загружаются в приложение, а запрашиваются онлайн с помощью веб-сервиса фирмы «1С». Для экономии трафика при автоподборе и проверке адресов, а также для автономной работы без (постоянного) подключения к Интернету адресные сведения могут быть загружены в приложение с сайта фирмы «1С» или из указанного каталога на компьютере.

Для загрузки адресных сведений из Интернета подсистему следует внедрять в конфигурацию совместно с подсистемой Получение файлов из Интернета. В противном случае доступна только загрузка адресных сведений из каталога на компьютере.

> **Важно**
>
> Подсистема доступна только в российской версии.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе ответственного за ведение нормативно-справочной информации необходимо разместить команды `ЗагрузитьАдресныйКлассификатор`, `ОчиститьАдресныйКлассификатор` регистра сведений `АдресныеОбъекты` для работы в варианте с загрузкой данных в приложение по требованию. См. пример в форме `ПоддержкаИОбслуживание` обработки `ПанельАдминистрированияБСП`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Адресный классификатор** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**)Просмотр и использование адресных сведений, загруженных в приложение.
2.	`ДобавлениеИзменениеАдресныхСведений` Загрузка и обновления адресных сведений из Интернета с сайта фирмы «1С» или из указанного каталоге на компьютере.
3.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Удаление помеченных на удаление объектов подсистемы.

Пример настройки прав доступа пользователей приведен ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор.	• `АдминистраторСистемы` (из подсистемы **Базовая функциональность**)
2.	Ответственный за нормативно-справочную информацию.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеАдресныхСведений`

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс. Доступ к данным адресного классификатора рекомендуется производить только через его программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы не должны включаться в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места. Данные адресного классификатора должны заполняться и поддерживаться в актуальном состоянии отдельно в каждой информационной базе, между которыми настроена синхронизация данных.

Анкетирование

Подсистема **Анкетирование** предназначена для составления анкет, проведения опросов и анализа результатов опросов. С помощью веб-клиента можно проводить опросы через Интернет.

Для аутентификации респондентов в системе используются возможности подсистемы **Пользователи**. Авторизовавшийся респондент получает доступ к предназначенным для него опросам, а также к архиву ранее заполненных им анкет.

Подсистема **Анкетирование** предоставляет ряд отчетов, которые позволяют анализировать в различных разрезах результаты одного выбранного опроса и сравнивать результаты, полученные при проведении нескольких опросов.

Настройка

Для того чтобы респонденты имели возможность указывать в качестве ответов на вопросы анкеты объекты информационной базы, необходимо принять решение по поводу состава типов таких объектов (определить типы ответов). Например, справочники **Номенклатура**, **Партнеры**, **Подразделения**. Список допустимых типов ответов надо добавить в состав типов свойства `Тип` значения ПВХ `ВопросыДляАнкетирования`.

Затем необходимо принять решение по поводу состава объектов конфигурации, которые могут быть сопоставлены с респондентами анкеты. Например, справочники **Контрагенты**, **Партнеры**, **Физические лица**. Список допустимых типов респондентов надо указать в составе типов определяемого типа `Респондент`.

Допустимые типы респондентов должны быть также включены в список внешних пользователей системы. Порядок действий по настройке подсистемы **Пользователи** описан в разделе Пользователи.

Для использования режима **Интервью** – заполнения анкет интервьюером за респондента – необходимо принять решение по поводу состава объектов конфигурации, которые могут быть сопоставлены с интервьюерами анкеты. Например, справочники **Пользователи**, **Внешние пользователи**, **Сотрудники**. Список допустимых типов интервьюеров надо указать в составе типов определяемого типа `Интервьюер`. В тех местах приложения, где необходимо обеспечить возможность начала анкетирования в режиме **Интервью**, потребуется вставить вызов экспортного метода `НачатьИнтервью` из модуля `АнкетированиеКлиент`.

Настройка пользовательского интерфейса

Для использования подсистемы в конфигурации необходимо настроить командный интерфейс исходя из функциональных обязанностей различных пользователей данной подсистемы.

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить константу `ИспользоватьАнкетирование`. См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

В командный интерфейс ответственного за составление анкет следует включить:

- ПВХ `ВопросыДляАнкетирования`,
- справочник `ШаблоныАнкет`.

В командном интерфейсе ответственного за проведение опросов необходимо разместить два документа:

- `НазначениеОпросов`,
- `Анкета`.

В командный интерфейс ответственного за анализ результатов анкетирования необходимо включить ряд отчетов:

- `АнализОпроса`,

- `АналитическийОтчетПоАнкетированию`.

Для респондента в рабочую область рабочего стола следует поместить основную форму обработки `ДоступныеАнкеты`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы `Анкетирование` следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ЧтениеОтветовНаВопросыАнкет` Просмотр всех анкет, опросов, ответов на вопросы анкет и анализ ответов в отчетах.
2.	`ДобавлениеИзменениеОтветовНаВопросыАнкет` Участие в опросах в качестве респондента или интервьюера. Доступны только те анкеты, в которых пользователь является интервьюером или респондентом.
3.	`ДобавлениеИзменениеШаблоновАнкет` Подготовка шаблонов анкет. Добавление и изменение вопросов для анкетирования.
4.	`ДобавлениеИзменениеОпросов` Проведение опросов.
5.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение использования подсистемы **Анкетирование**. Удаление помеченных на удаление объектов подсистемы.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным, которые не относятся к подсистеме **Анкетирование**, но требуются для работы с ней.

№	Вспомогательные роли и их назначение
1.	`<ЧтениеДанныхДляОтветовНаВопросыАнкет>` Просмотр объектов, значения которых могут выступать в качестве ответов на вопросы.
2.	`<ЧтениеДанныхРеспондентов>` Просмотр объектов, значения которых могут выступать в качестве респондентов.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор.	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Ответственный за анализ результатов анкетирования: • просмотр заполненных анкет, • использование аналитических отчетов подсистемы.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеВнешнихПользователей` (из подсистемы **Пользователи**), • `ЧтениеОтветовНаВопросыАнкет`, • `ЧтениеДанныхРеспондентов`, • `ЧтениеДанныхДляОтветовНаВопросыАнкет`.
3.	Составитель шаблонов анкет: • создание и редактирование шаблонов анкет и вопросов для анкетирования.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеШаблоновАнкет`, • `ЧтениеДанныхДляОтветовНаВопросыАнкет`.
4.	Ответственный за проведение опросов: • назначение опросов.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеВнешнихПользователей` (из подсистемы **Пользователи**), • `ДобавлениеИзменениеОпросов`, • `ЧтениеДанныхРеспондентов`.
5.	Респондент: • участие в опросах в качестве респондента.	• `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускВебКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеДанныхДляОтветовНаВопросыАнкет`.

Использование при разработке конфигурации

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Базовая функциональность

Подсистема **Базовая функциональность** содержит базовые функции, обязательные для всех прикладных решений, использующих библиотеку. К базовым относятся механизмы запуска и завершения работы конфигурации, процедуры и функции общего назначения, ряд универсальных обработок, а также стандартные роли: `ПолныеПрава`, `БазовыеПраваБСП` и другие.

Настройка

Базовые типы

Если в конфигурации имеется справочник `Организации`, то необходимо:

- указать ссылку на него в свойстве `Тип` определяемого типа `Организация`;
- реализовать в модуле менеджера справочника `Организации` в области `ПрограммныйИнтерфейс` две экспортные функции:

```
// Возвращает организацию по умолчанию.
// Если в ИБ есть только одна организация, которая не помечена на удаление и не является предопределенной,
// то будет возвращена ссылка на нее, иначе будет возвращена пустая ссылка.
//
// Возвращаемое значение:
//     ОпределяемыйТип.Организация - ссылка на организацию.
//
Функция ОрганизацияПоУмолчанию() Экспорт

КонецФункции
// Возвращает количество элементов справочника Организации.
// Не учитывает предопределенные и помеченные на удаление элементы.
//
// Возвращаемое значение:
//     Число - количество организаций.
//
Функция КоличествоОрганизаций() Экспорт

КонецФункции
```

Использование глобальных переменных

Вместо создания глобальных переменных в модулях управляемого и обычного приложения рекомендуется обращаться к общей глобальной переменной `ПараметрыПриложения`.

```
// СтандартныеПодсистемы

// Хранилище глобальных переменных.
//
// ПараметрыПриложения - Соответствие из КлючИЗначение:
//   * Ключ - Строка - имя переменной в формате "ИмяБиблиотеки.ИмяПеременной";
//   * Значение - Произвольный - значение переменной.
//
// Пример инициализации:
//   ИмяПараметра = "СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации";
//   Если ПараметрыПриложения[ИмяПараметра] = Неопределено Тогда
//     ПараметрыПриложения.Вставить(ИмяПараметра, Новый СписокЗначений);
//   КонецЕсли;
//
// Пример использования:
//   ПараметрыПриложения["СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации"].Добавить(...);
//   ПараметрыПриложения["СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации"] = ...;
Перем ПараметрыПриложения Экспорт;

// Конец СтандартныеПодсистемы
```

Реализация обработчиков событий модулей управляемого и обычного приложения

Для ряда обработчиков событий `ПередНачаломРаботыСистемы`, `ПриНачалеРаботыСистемы`, `ОбработкаПолученияФормыВыбораПользователейСистемыВзаимодействия` и других предусмотрена соответствующая процедура общего модуля `СтандартныеПодсистемыКлиент`. В таких обработчиках вызов процедуры общего модуля `СтандартныеПодсистемыКлиент` должен быть первым. Например:

```
Процедура ОбработкаПолученияФормыВыбораПользователейСистемыВзаимодействия(НазначениеВыбора,
            Форма, ИдентификаторОбсуждения, Параметры, ВыбраннаяФорма, СтандартнаяОбработка)

    // СтандартныеПодсистемы
    СтандартныеПодсистемыКлиент.ОбработкаПолученияФормыВыбораПользователейСистемыВзаимодействия(НазначениеВыбора,
        Форма, ИдентификаторОбсуждения, Параметры, ВыбраннаяФорма, СтандартнаяОбработка);
    // Конец СтандартныеПодсистемы

КонецПроцедуры
```

Если в общем модуле `ОбщегоНазначенияКлиентПереопределяемый` предусмотрен одноименный обработчик события, тогда следует делать вставку кода в него вместо вставки кода непосредственно в обработчик события модуля приложения.

```
Процедура ПриНачалеРаботыСистемы(Параметры) Экспорт

    Параметры.Модули.Добавить(ОбщегоНазначенияУТКлиент);
    ...
КонецПроцедуры

// См. ОбщегоНазначенияКлиентПереопределяемый.ПриНачалеРаботыСистемы
Процедура ПередНачаломРаботыСистемы(Параметры) Экспорт
    ...
КонецПроцедуры
```

Вставка кода должна быть в виде одной строки на одну библиотеку в формате `ИмяОбщегоМодуля.ИмяПроцедурыСобытия` с тем же составом параметров, что у события платформы, например:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные)

    МенеджерОборудованияКлиент.ОбработкаВнешнегоСобытия(Источник, Событие, Данные);

КонецПроцедуры
```

Инициализация параметров сеанса

Для инициализации параметров сеанса требуется вписать имя параметра сеанса и путь к его обработчику в процедуру `ПриДобавленииОбработчиковУстановкиПараметровСеанса` общего модуля `ОбщегоНазначенияПереопределяемый`. При этом обработчик инициализации должен принимать два параметра:

- `ИмяПараметра` – строка – имя инициализируемого параметра;
- `УстановленныеПараметры` – массив – имена параметров, которые были инициализированы.

Для действий при первом вызове события модуля сеанса `УстановкаПараметровСеанса` (`ИменаПараметровСеанса = Неопределено`) предусмотрен обработчик события `ПередЗапускомПрограммы` в общем модуле `ОбщегоНазначенияПереопределяемый`. Следует вписывать вызовы в этот обработчик вместо обработчика события платформы `УстановкаПараметровСеанса` в модуле сеанса.

Настройка исключений поиска ссылок на объекты

В тех случаях, когда ссылки между объектами информационной базы не являются значимыми для таких операций, как удаление объекта или поиск ссылок на объект, необходимо настроить список исключений поиска ссылок в функции `ПриДобавленииИсключенийПоискаСсылок` в переопределяемом модуле `ОбщегоНазначенияПереопределяемый`.

Список исключений поиска ссылок может также использоваться другими подсистемами или в функциональности конфигурации. См. также раздел Удаление помеченных объектов.

В частности, при анализе мест использования с помощью функции `ОбщегоНазначения.МестаИспользования` в результатах поиска ссылающихся объектов не учитываются ссылки из этого списка исключений. Использовать метод `НайтиПоСсылкам` в общем случае не рекомендуется.

Настройка пользовательского интерфейса

При запуске автоматически устанавливается заголовок окна приложения по значению функции `Пользователи.АвторизованныйПользователь` и значению константы `ЗаголовокСистемы`.

Пример заголовка системы:

Торговый дом «Ромашка» / Романова / Бухгалтерия предприятия, редакция 2.0 / (1С:Предприятие)

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить константу `ЗаголовокСистемы`. Для того чтобы изменения вступили в силу, требуется вызвать процедуру `УстановитьПроизвольныйЗаголовокПриложения` общего модуля `ОбщегоНазначенияКлиент`. См. пример в форме `ОбщиеНастройки` обработки `ПанельАдминистрированияБСП`.

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо дополнительно разместить:

- общую команду `УстановитьРасширениеДляРаботыС1СПредприятием`,
- обработку **Журнал регистрации**,

- команду для вызова формы `ПараметрыАдминистрированияСервернойИБ` в форме административных настроек системы.

См. пример размещения в формах обработки `ПанельАдминистрированияБСП`.

В форме персональных настроек следует разместить поле для изменения настройки, управляющей подтверждением при закрытии приложения, общую команду `УстановитьРасширениеДляРаботыС1СПредприятием`, а также другие персональные настройки подсистем. См. пример реализации в демонстрационной конфигурации в общей форме `_ДемоМоиНастройки`.

В случаях, когда конфигурация поддерживает работу в модели сервиса и есть объекты, видимость которых должна зависеть от текущего режима работы приложения, то их необходимо включить в состав следующих функциональных опций:

- `РаботаВАвтономномРежиме`.
- `РаботаВЛокальномРежиме`.
- `РаботаВМоделиСервиса`.

Подробнее см. в разделе Работа в модели сервиса.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Выполнение кода при запуске системы

Действия, выполняемые перед началом работы системы, при начале работы системы и перед завершением работы системы, следует располагать в процедурах общего модуля `ОбщегоНазначенияКлиентПереопределяемый`: `ПередНачаломРаботыСистемы`, `ПриНачалеРаботыСистемы` и `ПередЗавершениемРаботыСистемы` соответственно. При этом следует учитывать, что при работе в модели сервиса данные процедуры могут быть вызваны не только при осуществлении фактического входа (выхода) пользователя из системы, но и при интерактивном входе (выходе) администратора информационной базы в область данных.

С целью минимизации количества серверных вызовов при старте системы не рекомендуется напрямую вызывать серверные процедуры и функции из кода модуля приложения и модуля управляемого приложения. Для передачи клиенту параметров, необходимых для выполнения клиентского кода, следует использовать функцию `ПараметрыРаботыКлиентаПриЗапуске` общего модуля `СтандартныеПодсистемыКлиентПовтИсп`. При первом вызове этой функции происходит одно обращение к серверу, после чего полученное значение кешируется на клиенте для всех последующих вызовов этой функции.

При необходимости расширить количество передаваемых параметров следует добавлять новые параметры в функцию `ПараметрыРаботыКлиентаПриЗапуске` переопределяемого общего модуля `ОбщегоНазначенияПереопределяемый`. Новые параметры рекомендуется помещать после параметров библиотечных подсистем в произвольном порядке относительно друг друга, например:

```
Параметры.Вставить("ИнформационнаяБазаФайловая", ОбщегоНазначения.ИнформационнаяБазаФайловая());
```

Аналогичный подход рекомендуется применять и для взаимодействия с сервером при дальнейшей работе в приложении. См. функцию `ПараметрыРаботыКлиента` переопределяемого общего модуля `ОбщегоНазначенияПереопределяемый`.

Выполнение действий пользователя в фоне на сервере

Для выполнения действий пользователя, требующих продолжительной обработки данных (больше 2-3 секунды, например для формирования отчетов), следует руководствоваться стандартом Длительные операции на сервере. Для типового решения задачи реализован механизм длительных операций. Использование механизма описано в стандарте, а программный интерфейс размещен в общих модулях `ДлительныеОперации` и `ДлительныеОперацииКлиент`.

Обновление вспомогательных данных во время разработки

В ряде случаев при разработке и отладке конфигурации может потребоваться обновление вспомогательных данных, которые влияют на работу приложения: кэши некоторых свойств метаданных, служебные регистры сведений и т. п.

- Для перезаполнения этих данных (рекомендуется при разработке) предназначена внешняя обработка `ОбновлениеВспомогательныхДанных.epf`, которая входит в состав дистрибутива библиотеки.
- Для стандартного обновления (только в части изменений конфигурации) можно указать параметр запуска `ЗапуститьОбновлениеИнформационнойБазы` в конфигураторе или через параметр командной строки **/C**.
- При закладке изменений в хранилище, которые могут привести к необходимости обновления вспомогательных данных, можно увеличивать номер версии конфигурации, тогда у других участников коллективной разработки автоматически запустятся обязательные обработчики обновления.

Такие случаи специально отмечены в тексте документации соответствующих подсистем и выделены префиксом **Внимание**.

В остальных случаях обновление вспомогательных данных выполняется автоматически при каждом изменении номера версии конфигурации – в процессе обновления или первоначального заполнения информационной базы.

Хранение ссылок на объекты метаданных

При необходимости хранить в базе данных ссылку на объект метаданных (например, ссылка на объект метаданных `Справочник.Организации`) рекомендуется вместо строкового реквизита с полным именем объекта метаданных использовать составной тип со ссылками на справочники

`ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений`. Такая потребность возникает, например, для хранения настроек истории изменений справочников и документов, списка ролей в профилях групп доступа, размещения отчетов в разделах командного интерфейса и т. п.

Идентификаторы объектов метаданных позволяют:

- повысить производительность запросов, которые обращаются к реквизитам данного типа;
- снизить размер таблиц в базе данных, в которых используются реквизиты данного типа;
- выводить в пользовательском интерфейсе представление ссылки вместо строкового имени объекта метаданных;
- избавиться от разработки обработчиков обновления и первоначального заполнения ИБ для актуализации строковых реквизитов с полными именами объектов метаданных при изменениях метаданных конфигурации.

> **Важно!**
>
> Справочники `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений` не предназначены 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.

Программно ссылку на объект метаданных можно получить с помощью функций `ИдентификаторОбъектаМетаданных` и `ИдентификаторыОбъектовМетаданных` общего модуля `ОбщегоНазначения`.

Состав элементов справочника `ИдентификаторыОбъектовМетаданных` заполняется и актуализируется автоматически при первом запуске и каждом обновлении версии конфигурации согласно метаданным конфигурации: учитываются переименованные, добавленные и удаленные объекты метаданных. Например, если справочник `Товары` был переименован в новой версии конфигурации в справочник `Номенклатура`, то соответствующий элемент справочника сохранит прежнюю ссылку и будет переименован. Состав справочника `ИдентификаторыОбъектовРасширений` актуализируется автоматически при подключении и отключении расширений конфигурации.

Еще одним типовым случаем является переименование объектов метаданных с целью изменения структуры данных с помощью создания новой копии объекта метаданных. Такая необходимость возникает, когда реструктуризация объекта метаданных невозможна. Например, при сокращении длины кода справочника `Подразделения` с 50 до 11 может потребоваться создать новый справочник, а старый переименовать в `УдалитьПодразделения`. В этом случае существующий идентификатор справочника будет автоматически переназначен новому справочнику с тем же именем.

Однако если при этом новый справочник будет назван иначе, например `СтруктурныеЕдиницы`, то потребуется отразить переименование справочника `Подразделения` в справочник `СтруктурныеЕдиницы`, как это описано далее.

Переименование подсистем и двойные переименования объектов метаданных

Автоматически не отслеживаются следующие случаи переименования объектов метаданных:

- переименования подсистем[[1]](#прим1_1);
- перемещение подсистемы из одной родительской подсистемы в другую;
- двойные переименования, например: справочник `Подразделения` > `УдалитьПодразделения` + `СтруктурныеЕдиницы`.

Эти случаи требуется описать в явном виде в процедуре `ПриДобавленииПереименованийОбъектовМетаданных` общего модуля `ОбщегоНазначенияПереопределяемый` и увеличить номер версии конфигурации. Например, следующий фрагмент кода описывает, что в версии конфигурации 2.0.1.2 подсистема `_ДемоПоставляемыеДанные` была перенесена из подсистемы `_ДемоРаботаВМоделиСервиса` в `_ДемоАдминистрирование`:

```
Процедура ПриДобавленииПереименованийОбъектовМетаданных(Итог) Экспорт
    ОбщегоНазначения.ДобавитьПереименование(Итог, "2.0.1.2",
        "Подсистема._ДемоРаботаВМоделиСервиса.Подсистема._ДемоПоставляемыеДанные",
        "Подсистема._ДемоАдминистрирование.Подсистема._ДемоПоставляемыеДанные");
КонецПроцедуры
```

> [1] Также необходимо указывать переименования ролей в тех случаях, когда необходимо обеспечить обновление с предыдущих версий конфигурации, в которые включена библиотека версии 3.1.5 и меньше.

Обновление идентификаторов выполняется последовательно по версиям конфигурации, а в пределах одной версии – в порядке следования строк со сведениями о переименованиях. Обновление идентификаторов для переименованных подсистем выполняется также и для всех их дочерних подсистем (если они есть).

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) не следует переименовывать роли и подсистемы в младших версиях. Правильно выполнять такие переименования только в самой последней разрабатываемой версии (которая еще не выпущена).

Например, неправильно: в исправительных релизах версий 2.4.5, 3.0.1 переименовали подсистему `НачальнаяСтраница` в `Главное`.

Правильно выполнять такое переименование только в самой последней разрабатываемой версии 3.0.2, а в исправительных релизах ограничиться переименованием синонима.

В противном случае, при переходе с младшей версии на старшую версию это переименование может быть учтено дважды, что приведет к рассогласованию ссылок на объекты метаданных и к различным ошибкам в тех подсистемах конфигурации, которые на них опираются. 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.

Удаление идентификаторов объектов метаданных и скрытие их в пользовательском интерфейсе

При удалении объектов метаданных из конфигурации и при удалении расширений конфигурации из информационной базы, как правило, больше не требуется хранить связанные с ними данные. Удаление этих данных информационной базы, связанных с удаленными объектами метаданных конфигурации и расширений, выполняется централизованно с помощью удаления помеченных объектов (см. соответствующий раздел документации). При этом не рекомендуется выполнять очистку в другие моменты времени, например, при записи объектов, так как это может вызывать избыточную нагрузку на систему.

Например, регистр `НастройкиСинхронизацииДанных` хранит определенные настройки в разрезе объектов метаданных. Для этого в его ведущем измерении указан составной тип – ссылки на справочники `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений`. При удалении объекта метаданных соответствующий ему элемент справочника `ИдентификаторыОбъектовМетаданных` автоматически помечается на удаление, а затем, при удалении помеченных объектов, записи регистра `НастройкиСинхронизацииДанных`, связанные с этим элементом, автоматически удаляются средствами платформы **1С:Предприятие** (т.к. измерение регистра отмечено как ведущее).

В иных случаях для реализации очистки, когда ссылки на эти справочники хранятся в табличной части объекта или не в ведущих измерениях регистров, следует дополнительно реализовать подписку `ПередУдалением` для справочников `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений`, а сами реквизиты со ссылками отметить как исключения для поиска ссылок (для того чтобы они не препятствовали удалению). См. например подписку на событие `ВариантыОтчетовПередУдалениемИдентификатораОбъектаМетаданных` и процедуру `ВариантыОтчетов`. `ПриДобавленииИсключенийПоискаСсылок` в подсистеме **Варианты отчетов**.

Также рекомендуется скрывать в пользовательском интерфейсе данные, связанные с отсутствующими объектами метаданных (например, когда объект метаданных удален из конфигурации или администратор отключил соответствующее расширение конфигурации), а также корректно обрабатывать (пропускать) такие записи в алгоритмах обработки данных. Для этого в функциях `ОбъектМетаданныхПоИдентификатору` и `ОбъектыМетаданныхПоИдентификаторам` общего модуля `ОбщегоНазначения` нужно задавать параметру `ВызыватьИсключение` значение `Ложь`. Это позволяет в вызывающем коде различать ссылки на:

- удаленные объекты метаданных конфигурации;
- метаданные расширений, которые не подключены в текущем сеансе (администратор временно отключил расширение или требуется перезапуск сеанса);
- и действительные (существующие) объекты метаданных, с которыми допустимо вести работу.

Принудительное обновление идентификаторов объектов метаданных

При активной разработке конфигурации данные справочника `ИдентификаторыОбъектовМетаданных` можно также обновлять без увеличения номера версии конфигурации с помощью обработки `ОбновлениеВспомогательныхДанных.epf` (входит в состав дистрибутива библиотеки - см. раздел Обновление вспомогательных данных во время разработки). Однако это допустимо только в случае автоматически отслеживаемых случаев, например, когда не производилось переименование ролей или подсистем. В противном случае необходимо увеличить версию конфигурации и вписать переименование в процедуру `ПриДобавленииПереименованийОбъектовМетаданных`, как написано выше.

В нештатных случаях также можно устранить рассогласование идентификаторов объектов метаданных с объектами метаданных конфигурации, открыв форму соответствующего элемента справочника `ИдентификаторыОбъектовМетаданных`, выполнить команду `Включить возможность редактирования` (меню **Еще**) и задать корректное полное имя.

Вспомогательные предопределенные элементы для RLS

При использовании подсистемы **Управление доступом** в справочниках `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений` также должны быть созданы вспомогательные предопределенные элементы, которые используются в шаблонах ограничения доступа (RLS). Проверять и обновлять их состав рекомендуется автоматически с помощью отчета `ПроверкаВнедренияБСП.erf`. Подробнее см. раздел Регулярный запуск отчета ПроверкаВнедренияБСП.

Версионирование программных интерфейсов

При разработке нескольких конфигураций, которые должны взаимодействовать друг с другом (например, с помощью веб-сервисов или обмена сообщениями), как правило, на стороне каждой из конфигураций реализуется программный интерфейс, к которому обращается другая конфигурация. По мере развития каждой из конфигураций часто возникает необходимость внесения изменений в программные интерфейсы, предоставляемые этими конфигурациями своим корреспондентам.

В общем случае при изменении реализации программного интерфейса (далее – просто интерфейса), например набора методов или структуры их параметров, код вызывающей стороны становится неработающим, поскольку поведение вызывающих методов «жестко» завязано на конкретную реализацию интерфейса.

Для исключения коллизий с некорректным использованием измененного интерфейса следует использовать механизм версионирования интерфейсов.

Определения:

- Семейство интерфейсов – набор интерфейсов, решающих одну задачу и представленных в развитии реализации именами объектов метаданных с постфиксами номеров версий.
- Базовое имя интерфейсов семейства (или просто «базовое имя») – имя его представителя без постфиксного номера версии.

- Номер версии состоит из четырех групп, разделенных точками. Каждая из этих групп может содержать исключительно числовое значение. Например, **2.0.1.6**.

Приращение номеров версий осуществляется стандартным для версионирования ПО образом. Например, **1.0.2.1**, **1.0.2.2** … **1.0.3.1** и т. д.

- Интерфейс, носящий базовое имя, считается интерфейсом версии **1.0.1.1**.

Действия на стороне, предоставляющей интерфейс (например, веб-сервис `MessageExchange`):

- Создать интерфейс семейства с новой реализацией, наименовав его по правилу: **Базовое имя + «_» + Номер версии с подчеркиваниями вместо точек**. Например, `MessageExchange_2_0_1_6`.
- Присвоить семейству интерфейсов имя. Например, `ОбменСообщениями`. Имя семейству интерфейсов присваивается произвольно, исходя из его назначения.
- В теле процедуры `ПриОпределенииПоддерживаемыхВерсийПрограммныхИнтерфейсов` общего модуля `ОбщегоНазначенияПереопределяемый` к секции подсистемы интерфейса добавить вставку нового номера версии, как показано в примере описания процедуры. Если секция отсутствует, необходимо ее создать.

Действия на стороне, использующей интерфейс:

- Механизм версионирования обслуживается веб-сервисом `InterfaceVersion`, доступ к которому предоставляет роль `УдаленныйДоступБазоваяФункциональность`. Таким образом, функционирование механизма возможно только для пользователя, обладающего этой ролью.
- Перед вызовом методов интерфейса публикующей стороны нужно запросить у нее список номеров поддерживаемых версий через вызов функции `ПолучитьВерсииИнтерфейса` общего модуля `ОбщегоНазначения`, как показано в примере ее описания. При этом в качестве параметра `ИмяИнтерфейса` нужно задать имя семейства интерфейсов.
- В зависимости от полученного набора номеров версий в логических ветках осуществлять вызовы тех или иных методов с актуальной структурой параметров, обращаясь к конкретным интерфейсам семейства.

Примеры использования можно найти в демонстрационной конфигурации (см. вызовы функции `ПолучитьВерсииИнтерфейса`).

Безопасное хранилище паролей

Для хранения паролей и другой конфиденциальной информации следует использовать безопасное хранилище. Для помещения данных в безопасное хранилище предназначена процедура `ЗаписатьДанныеВБезопасноеХранилище`, для получения ранее сохраненных данных – функция `ПрочитатьДанныеИзБезопасногоХранилища`, а для удаления данных – функция `УдалитьДанныеИзБезопасногоХранилища` общего модуля `ОбщегоНазначения`.

Если в форме необходимо разместить поле для ввода пароля, то необходимо:

- Создать реквизит формы с типом `Строка`, включить свойство **Сохраняемые данные**, а также вспомогательный реквизит с типом `Булево` для фиксации факта изменения пароля. Например, `Пароль` и `ПарольИзменен`.

- Создать на форме поле со свойством `РежимПароля = Да` и связать его с ранее созданным реквизитом формы.

- Для сведения к минимуму возможности получения пароля злоумышленниками не следует при открытии формы присваивать его реквизиту формы и передавать с сервера на клиент. Вместо этого в событие формы `ПриСозданииНаСервере` на сервере надо разместить следующий код:

```
УстановитьПривилегированныйРежим(Истина);
Пароли = ОбщегоНазначения.ПрочитатьДанныеИзБезопасногоХранилища(Объект.Ссылка, "Пароль, ПарольSMTP");
УстановитьПривилегированныйРежим(Ложь);
Пароль = ?(ЗначениеЗаполнено(Пароли.Пароль), УникальныйИдентификатор, "");
ПарольSMTP = ?(ЗначениеЗаполнено(Пароли.ПарольSMTP), УникальныйИдентификатор, "");
```

- В событие `ПриЗаписиНаСервере`:

```
Если ПарольИзменен Тогда
    УстановитьПривилегированныйРежим(Истина);
    ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, Пароль);
    УстановитьПривилегированныйРежим(Ложь);
КонецЕсли;
Если ПарольSMTPИзменен Тогда
    УстановитьПривилегированныйРежим(Истина);
    ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(ТекущийОбъект.Ссылка, ПарольSMTP, "ПарольSMTP");
    УстановитьПривилегированныйРежим(Ложь);
КонецЕсли;
```

- В модуле объекта разместить следующий код в событие `ПередУдалением`:

```
УстановитьПривилегированныйРежим(Истина);
ОбщегоНазначения.УдалитьДанныеИзБезопасногоХранилища(Ссылка);
УстановитьПривилегированныйРежим(Ложь);
```

См. пример в форме `ФормаЭлемента` справочника `УчетныеЗаписиЭлектроннойПочты`.

Отображение исключений для пользователей

При работе приложения могут возникать исключения, которые вызывает платформа **1С:Предприятие** или прикладной код с помощью метода `ВызватьИсключение`. Необработанные исключения автоматически записываются в журнал регистрации и выводятся в стандартном окне, в котором даны типовые рекомендации для пользователя и предусмотрена гиперссылка для отправки разработчику отчета об ошибке (в зависимости от категории ошибки). Для того чтобы придерживаться этого подхода, рекомендуется:

1. Возвращать и отображать ошибки длительных операций, которые запускаются функциями `ВыполнитьФункцию`, `ВыполнитьПроцедуру`, `ВыполнитьФункциюВНесколькоПотоков`, `ВыполнитьПроцедуруВНесколькоПотоков` и `ВыполнитьВФоне` общего модуля `ДлительныеОперации`:

- При возникновении исключения в длительной операции в обработчик результата передается структура `ДлительныеОперацииКлиент.НовыйРезультатДлительнойОперации` с заполненным свойством `ИнформацияОбОшибке` (тип `ИнформацияОбОшибке`). Для отображения этой ошибки рекомендуется вызывать процедуру `СтандартныеПодсистемыКлиент.ВывестиИнформациюОбОшибке` вместо показа предупреждения или вызова метода платформы `ПоказатьИнформациюОбОшибке`. Данная процедура требуется для перехвата исключений системами автоматического тестирования (так как при вызове метода платформы `ПоказатьИнформациюОбОшибке` не срабатывает обработчик события `ОбработкаОтображенияОшибки` модуля приложения).
- В ряде случаев возникает необходимость дополнить текст ошибки длительной операции определенным контекстом, например: "Не удалось <выполнить действие> по причине: ...". В таких случаях рекомендуется добавлять подобное уточнение не в клиентском коде, а сразу передавать текст уточнения в свойстве `УточнениеОшибки` в параметрах функций `ПараметрыВыполненияФункции`, `ПараметрыВыполненияПроцедуры` и `ПараметрыВыполненияВФоне` общего модуля `ДлительныеОперации`. Например:

  ```
  ПараметрыВыполнения.УточнениеОшибки = НСтр("ru = 'Не удалось <выполнить действие> по причине:'");
  ```

  Тем самым, сообщение, выводимое пользователю, будет в точности совпадать с информацией об ошибке, которая записывается в журнал регистрации.
- Даже если в процедуре длительной операции возникающие исключения перехватываются и подробное представление ошибки записывается в журнал регистрации, рекомендуется пробрасывать исключения и отображать их пользователю объект `ИнформацияОбОшибке` в стандартном виде, как указано выше.

1. В тех случаях, когда для отображения ошибок реализован собственный пользовательский интерфейс, также рекомендуется придерживаться стандартного поведения платформы:

- Выводить в пользовательском интерфейсе результат метода `ОбработкаОшибок.СообщениеОбОшибкеДляПользователя(ИнформацияОбОшибке)` в виде форматированной строки и обеспечить обеспечить просмотр и отправку отчета по гиперссылке **Сформировать отчет об ошибке**:

```
ОтчетДляОтправки = Новый ОтчетОбОшибке(ИнформацияОбОшибке);
СтандартныеПодсистемыКлиент.ПоказатьОтчетОбОшибке(ОтчетДляОтправки);
```

- При этом для управления видимостью гиперссылки в зависимости от категории ошибки и настроек приложения рекомендуется вызывать процедуру `СтандартныеПодсистемыКлиент.НастроитьВидимостьИЗаголовокСсылкиОтправкиОтчетаОбОшибке` из обработчика модуля формы `ПриОткрытии`.
- В обработчике модуля формы `ПриЗакрытии` вызвать процедуру `СтандартныеПодсистемыКлиент.ОтправитьОтчетОбОшибке(ОтчетДляОтправки, ИнформацияОбОшибке)`, которая автоматически отправляет отчет, если ошибка соответствующей категории и в приложении настроена обязательная отправка.
- Пример реализации см. в формах `Обработка.РезультатыОбновленияПрограммы.Форма.СообщениеОНеудачномОбновлении`, `Обработка.ЗаменаИОбъединениеЭлементов.Форма.ЗаменаЭлементов`, `Обработка.ТекущиеДела.Форма.Форма`.

Стандартные роли и дополнительные права

Библиотека не навязывает ту или иную методику разработки ролей в конфигурации. В общем виде при разработке системы ролей могут использоваться два подхода, которые различаются степенью детализации (укрупненности) ролей:

1. При первом подходе проектируются **прикладные роли**, предоставляющие доступ ко всему множеству объектов метаданных, которые требуются для работы определенной категории пользователей системы. Например, **Бухгалтер**, **Кассир** и т. д. Такие роли самостоятельно назначают пользователям (группам пользователей) и, как правило, расширяют дополнительными правами, например: **Действия главного бухгалтера**, **Печать непроведенных документов**, **Запуск тонкого клиента** и т. п.
2. Во втором случае проектируются **роли-функции**, предоставляющие «атомарный» доступ к определенному подмножеству объектов метаданных, с которым различные пользователи могут работать как с одной функцией системы. Такие роли не назначаются пользователям поодиночке, а объединяются в профили групп доступа и назначаются пользователям (группам пользователей) в совокупности. При этом профили групп доступа выступают аналогами прикладных ролей, например: **Бухгалтер**, **Кассир** и т. д.

В состав библиотеки входят роли, которые можно использовать для настройки доступа пользователей к объектам информационной базы в обоих перечисленных случаях.

Базовые роли

№	Роли и их назначение
1.	`ПолныеПрава` Предоставляет неограниченный доступ ко всем данным приложения, но не дает прав доступа для администрирования информационной базы в целом (обновление конфигурации, работа в конфигураторе и т. п.). Включает все права доступа, кроме права интерактивного удаления. При работе в модели сервиса назначается администраторам абонентов (областей данных) и предоставляет неограниченный доступ ко всем данным текущей области, а также позволяет выполнять администрирование пользователей, настройку приложения, удаление помеченных объектов и другие административные действия с областью данных. При работе в локальном режиме назначается совместно с ролью `АдминистраторСистемы` для администрирования. В базовых версиях конфигурации роль `ПолныеПрава` предоставляет неограниченный доступ ко всем данным и конфигурации информационной базы.
2.	`АдминистраторСистемы` Предоставляет права администрирования информационной базы в целом (обновление конфигурации, работа в конфигураторе и т. п.). Обязательно назначается совместно с ролью `ПолныеПрава`. При работе в модели сервиса назначается администраторам сервиса. Включает неограниченный доступ ко всем неразделенным (общим) данным, кроме права интерактивного удаления. В базовых версиях конфигурации роль `АдминистраторСистемы` не используется.
3.	`Администрирование` Предоставляет права **Администрирование**, **Администрирование данных** и **Активные пользователи**. Подробнее о них см. документацию к платформе 1С:Предприятие.
4.	`БазовыеПраваБСП` Минимальные права к тем функциям приложения, которые должны быть доступны всем пользователям. Назначается всем пользователям.
5.	• `БазовыеПраваВнешнихПользователейБСП`. Минимальные права к тем функциям приложения, которые должны быть всегда доступны всем внешним пользователям. Назначается всем внешним пользователям.
6.	`ВыводНаПринтерФайлБуферОбмена` Вывод любых данных на печать, сохранение в файл на компьютере и копирование в буфер обмена (право **Вывод**).
7.	`ЗапускAutomation` Разрешает подключение к приложению из других приложений **1С:Предприятие** или сторонних приложений с помощью Windows-технологии Automation Server (право **Automation**).
8.	`ЗапускВебКлиента` Разрешает вход в приложение с помощью веб-клиента (право **Веб-клиент**).
9.	`ЗапускВнешнегоСоединения` Разрешает подключение к приложению из других приложений **1С:Предприятие** или сторонних приложений с помощью Windows-технологии COM (право **Внешнее соединение**).
10.	`ЗапускМобильногоКлиента` Разрешает вход в приложение со смартфонов, планшетов и других мобильных устройств с помощью мобильного клиента (право **Мобильный клиент**). Это устаревший вид клиентского приложения, поэтому вместо этой роли рекомендуется назначать роли `ЗапускТонкогоКлиента` или `ЗапускВебКлиента`.
11.	`ЗапускТолстогоКлиента` Разрешает вход в приложение с помощью толстого клиента (право **Толстый клиент**). Это устаревший вид клиентского приложения, поэтому вместо этой роли рекомендуется назначать роли `ЗапускТонкогоКлиента` или `ЗапускВебКлиента`.
12.	`ЗапускТонкогоКлиента` Разрешает вход в приложение с помощью тонкого клиента (право **Тонкий клиент**).
13.	• `ИнтерактивноеОткрытиеВнешнихОтчетовИОбработок`. Разрешает открывать внешние отчеты и обработки (права **Интерактивное открытие внешних отчетов** и **Интерактивное открытие внешних обработок**).
14.	`ОбновлениеКонфигурацииБазыДанных` Разрешает обновление конфигурации информационной базы (право **Обновление конфигурации базы данных**).
15.	`ПросмотрЖурналаРегистрации` Просмотр журнала регистрации (право **Журнал регистрации**).
16.	`РежимТехническогоСпециалиста` Управляет доступностью в меню Сервис и настройки (главном меню) команды Функции для технического специалиста и соответствующего пункта настроек клиентского приложения (право **Режим технического специалиста**).
17.	`СохранениеДанныхПользователя` • Возможность сохранения настроек пользователя (право **Сохранение данных пользователя**).

№	Роли и их назначение
18.	• `УдаленныйДоступБазоваяФункциональность`. • Разрешает вызовы операций веб-сервисов подсистемы **Базовая функциональность** из других приложений.

Подробнее о стандартных правах доступа см. документацию к платформе 1С:Предприятие.

Подход № 1. Прикладные роли и дополнительные права

В простейшем случае роль соответствует функциональным обязанностям определенной категории пользователей системы, например: **Бухгалтер**, **Кассир**. Такие прикладные роли применимы в конфигурациях, в которых функциональные обязанности пользователей заранее известны и не предполагают изменений на внедрениях.

Совместно с прикладными ролями пользователям могут назначаться одна или несколько ролей, предоставляющих дополнительные права. Название роли, предоставляющей дополнительные права на определенные действия, должно описывать эти действия, например: **Действия главного бухгалтера**, **Печать непроведенных документов**, **Запуск тонкого клиента**, **Удаление помеченных объектов** и т. п. Не следует применять в именах ролей слово «право» (исключения составляют обязательные роли `ПолныеПрава`, `БазовыеПраваБСП` и `БазовыеПраваВнешнихПользователейБСП`).

Роли, предоставляющие дополнительные права, не предназначены для самостоятельного использования, они применяются в комбинации с прикладной ролью и служат для расширения доступа к данным той или иной прикладной области. Например:

- роль **Действия генерального директора** может назначаться пользователям только совместно с «прикладной» ролью **Генеральный директор**;
- роль **Действия главного бухгалтера** – только вместе с ролью **Бухгалтер**;
- роль **Печать непроведенных документов** может использоваться совместно с произвольной «прикладной» ролью.

Перед началом работы конфигурации следует проверить состав ролей текущего пользователя. Если пользователю не назначена хотя бы одна прикладная роль, предполагающая самостоятельное использование, работа пользователя не может быть продолжена.

Если в конфигурации не используется подсистема **Управление доступом**, то для упрощения администрирования прав пользователей рекомендуется оставить в конфигурации только обязательные роли, удалив все остальные библиотечные роли.

При использовании подсистемы **Управление доступом** допускается оставить в конфигурации все библиотечные роли как есть и рекомендуется подготовить типовые профили групп доступа пользователей. Подробнее о настройке профилей групп доступа с помощью возможностей подсистемы **Управление доступом** (см. в разделе Управление доступом).

Подход № 2. Роли-функции

Если функциональные обязанности пользователей конфигурации неизвестны на этапе разработки, а определяются на конкретном внедрении, прикладные роли следует разделять на более специализированные роли, декомпозируя их до уровня отдельных ролей-функций.

Роли-функции не предназначены для самостоятельного использования, а назначаются пользователям (группам пользователей) всегда в комбинации с другими ролями-функциями. Совокупность ролей-функций, предоставляющих определенной категории пользователей необходимый набор прав для выполнения своих обязанностей, образует профиль группы доступа. Например, профиль **Менеджер по продажам** может включать такие роли-функции, как **Добавление и изменение заказов покупателей** и **Чтение нормативно-справочной информации**, а **Начальник отдела продаж** – **Добавление и изменение заказов покупателей**, **Добавление и изменение нормативно-справочной информации** и **Печать непроведенных документов**.

Настройка прав пользователей с помощью отдельных ролей-функций не требует внесения изменений в конфигурацию как на этапе внедрения, так и на этапе эксплуатации (администрирования) информационной системы.

При проектировании роли-функции нужно учитывать, что она участвует в формировании пользовательского интерфейса. Поэтому для достижения гибкости управления пользовательским интерфейсом могут потребоваться «атомарные» роли, которые предоставляют доступ к одному объекту метаданных.

Так, в случае с объектами метаданных, которые используются в качестве разрезов ограничения доступа (например, организаций, складов, контрагентов, касс и др.), для чтения лучше создавать «атомарные роли», т. к. они будут использоваться в сочетании с ролями – например, для доступа к документам. При таком подходе, например, в интерфейс кладовщика не попадет доступ к списку касс, который ему не требуется, что соответствует принципу управляемого интерфейса.

Для реализации подхода № 2 рекомендуется внедрение в конфигурацию подсистемы **Управление доступом**. Подробнее о настройке профилей групп доступа с помощью возможностей подсистемы **Управление доступом** (см. в разделе Управление доступом).

Требования к наименованию ролей-функций

Название роли-функции рекомендуется начинать с описания действия, которое она позволяет выполнить пользователю в информационной системе:

- `Чтение` – для права чтения и просмотра, например: `ЧтениеЗаказов`, `ЧтениеНормативноСправочнойИнформации` и т. п.;
- `ДобавлениеИзменение` – для права добавления, которое должно сочетаться с правом изменения добавленного, правами чтения и просмотра, например `ДобавлениеИзменениеЗаказов`;
- `Изменение` – для права изменения, например `ИзменениеЗадач`;

- `Редактирование` – для права интерактивного редактирования, которое должно сочетаться с правом изменения того, что редактируется, например `РедактированиеЗадач`;
- `Просмотр` – для права интерактивного просмотра, которое должно сочетаться с правом чтения того, что просматривается, например `ПросмотрПартнеров`;
- `ПометкаУдаления` – для права интерактивной установки пометки удаления, которое должно сочетаться с правом изменения того, что помечается на удаление, например `ПометкаУдаленияБизнесПроцессов`.

Если роль дает права сразу на несколько действий, то в синониме роли их следует перечислять через запятую и соединять союзом «и», например: **Чтение заказов покупателей**, **Добавление и изменение нормативно-справочной информации**, **Добавление, изменение и пометка удаления заказов покупателей**.

Управление командным интерфейсом с помощью ролей-функций

Видимость элементов пользовательского интерфейса в зависимости от прав пользователей рекомендуется задавать в соответствующих ролях-функциях. Например, в роль **Чтение заказов покупателей** следует включать общие команды, которые открывают формы заказов покупателей.

Однако в некоторых случаях все же необходимо заводить дополнительные роли для управления командным интерфейсом. Как правило, такие интерфейсные роли не содержат ограничений доступа к данным, а связываются непосредственно с реквизитами объектов, элементами форм и командами. Их наличие у текущего пользователя может также программно проверяться из кода на встроенном языке.

Название интерфейсной роли должно описывать разрешенное действие, например **Просмотр контактной информации**. Исключение составляют роли, которые предоставляют доступ к разделам командного интерфейса (просмотр подсистем). Такие роли начинаются со слова «Раздел», например: `РазделЗапасыИЗакупки`, `РазделПродажи`, `РазделПродажиИПроведениеСделок`, `РазделПродажиИВозвраты`, `РазделПродажиРозничныеПродажи`.

Примеры интерфейсных ролей приведены ниже.

№	Роли и их назначение
1.	**Раздел нормативно-справочная информация**. Роль управляет видимостью раздела командного интерфейса, в котором размещены команды открытия форм объектов метаданных, относящихся к нормативно-справочной информации, например: форма списка справочника **Валюты**, адресного классификатора и т. п.
2.	**Раздел совместная работа**. Роль управляет видимостью раздела командного интерфейса, используемого для работы с задачами и бизнес-процессами.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор: • имеет полный доступ к системе, • выполняет настройку прав (состав ролей) и ограничений доступа.	• `АдминистраторСистемы`, • `ПолныеПрава`.
2.	Ответственный за нормативно-справочную информацию.	• `ЗапускТонкогоКлиента`, • `БазовыеПраваБСП`, • `ДобавлениеИзменениеАдресныхСведений`, • `ДобавлениеИзменениеБанков`, • `ДобавлениеИзменениеКалендарныхГрафиков`, • `ДобавлениеИзменениеДополнительныхРеквизитовИСведений`, • `ДобавлениеИзменениеКурсовВалют`, • `ДобавлениеИзменениеГрафиковРаботы`, • `ДобавлениеИзменениеВидовКонтактнойИнформации`, • `Подсистема_ДемоНормативноСправочнаяИнформация`.

Роли расширений

Подходы разработки ролей расширений такие же, как для ролей конфигурации. Для упрощения администрирования следует придерживаться пункта **Стандартные роли расширений** стандарта **Стандартные роли**.

Контроль замены ссылок при записи объектов и выполнение связанных действий

При поиске и удалении дублей, программном вызове `ЗаменитьСсылки` общего модуля `ОбщегоНазначения` (и в любых других случаях замены ссылок в документах) документы записываются без перепроведения, чтобы не нарушать последовательности. Как следствие, не срабатывает событие `ОбработкаПроведения` и вся прикладная логика проведения по регистрам. Однако в некоторых случаях при замене ссылок может потребоваться выполнять часть этой логики. Для этого в события записи объектов (`ПередЗаписью` и `ПриЗаписи`) через `ДополнительныеСвойства` передаются параметры:

- `ЗаменаСсылок` – `Булево` – `Истина`. Позволяет определить, что отработал механизм замены ссылок.
- `ВыполненныеЗамены` – `Массив`. Описание выполненных замен. Элементы типа `Структура` с полями:

- `СсылкаДубля` — `Произвольный`. Элемент, который был заменен.
- `СсылкаОригинала` — `Произвольный`. Элемент, на который выполнена замена.
- `ВидРеквизита` — `Строка`. Вид реквизита, в котором выполнена замена. Может принимать следующие значения: `Реквизиты`, `СтандартныеРеквизиты`, `ТабличныеЧасти`, `СтандартныеТабличныеЧасти`, `Движения`, `Последовательности`, `НаборЗаписей`.
- `ИмяРеквизита` — `Строка`. Имя реквизита или имя табличной части.
- `Индекс` — `Число`, `Неопределено`. Индекс строки, в которой выполнена замена дубля на оригинал. Заполняется в случае, когда замена выполнена в табличной части.
- `ИмяКолонки` — `Строка`, `Неопределено`. Имя колонки, в которой выполнена замена дубля на оригинал. Заполняется в случае, когда замена выполнена в табличной части.

Эти параметры можно использовать для того, чтобы определить, «где» и «что именно было изменено», и при необходимости выполнить связанные действия. Например, создать ключи аналитики и выполнить их замену в регистрах.

Настройка рабочей даты пользователя

Довольно часто на практике информация в учетные системы вводится не сразу в момент совершения операции, а позже по времени (задним числом). Например, фактическое поступление на склад товаров от поставщика может произойти раньше, чем будут получены соответствующие финансовые документы. Задним числом чаще всего работают бухгалтеры, отражающие уже свершившиеся ранее факты хозяйственной жизни.

Поэтому для удобства пользователей разработчик конфигурации может предусмотреть возможность настройки даты, которую необходимо подставлять в новые документы, – так называемой «рабочей даты».

В общем модуле `ОбщегоНазначения` расположены функции для программного управления такой настройкой:

- процедура `УстановитьРабочуюДатуПользователя`,
- функция `РабочаяДатаПользователя`,
- функция `ТекущаяДатаПользователя`.

Пример их использования см. в демонстрационной конфигурации в модуле общей формы `_ДемоМоиНастройки`. С помощью переключателя на форме можно выбрать в качестве даты создания документа текущую дату компьютера либо иную указанную дату. Сведения о рабочей дате сохраняются в хранилище общих настроек информационной базы, для доступа к которому необходима роль `СохранениеДанныхПользователя`.

При создании новых документов для заполнения их даты из настройки пользователя можно использовать подписки на события `ОбработкаЗаполнения` и `ПриКопировании` (см. в демонстрационной конфигурации подписки на события `_ДемоЗаполнитьДатуДокументаПоРабочейДате` и `_ДемоЗаполнитьДатуДокументаПоРабочейДатеПриКопировании`).

При использовании подписок на события необходимо учитывать, что их обработчики вызываются после того, как будет выполнен обработчик события в модуле объекта. Поэтому если алгоритм автоматического заполнения данных нового документа использует его дату, то установку даты документа из настройки рабочей даты следует выполнить в самом начале процедуры – обработчика события.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме перечисленных ниже:

- константа `АдресПубликацииИнформационнойБазыВИнтернете`,
- константа `АдресПубликацииИнформационнойБазыВЛокальнойСети`,
- константа `ДоставлятьСерверныеОповещенияБезСистемыВзаимодействия`,
- константа `ГлавныйУзел`,
- константа `ЗаголовокСистемы`,
- константа `ИдентификаторИнформационнойБазы`,
- константа `ИспользоватьРазделениеПоОбластямДанных`,
- константа `НеИспользоватьРазделениеПоОбластямДанных`,
- константа `РегистрироватьПоказателиСерверныхОповещений`,
- константа `СостояниеОтправкиСерверныхОповещений`,
- константа `ЭтоАвтономноеРабочееМесто`,
- справочник `ВерсииРасширений`,
- справочник `ИдентификаторыОбъектовРасширений`,
- регистр сведений `БезопасноеХранилищеДанных`,
- регистр сведений `БезопасноеХранилищеДанныхОбластейДанных`,
- регистр сведений `ЗапросыРазрешенийНаИспользованиеВнешнихРесурсов`,
- регистр сведений `ИдентификаторыОбъектовВерсийРасширений`,
- регистр сведений `КэшПрограммныхИнтерфейсов`,
- регистр сведений `ОтправленныеСерверныеОповещения`,
- регистр сведений `ПараметрыРаботыВерсийРасширений`,
- регистр сведений `ПериодическиеСерверныеОповещения`,
- регистр сведений `СвойстваРасширений`.

В планах обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется отключать регистрацию изменений для следующих объектов метаданных подсистемы (см. также раздел Особенности создания начального образа подчиненного узла распределенной ИБ):

- регистр сведений `ПараметрыРаботыПрограммы`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется исключать следующие объекты метаданных:

- константа `ДоставлятьСерверныеОповещенияБезСистемыВзаимодействия`,
- константа `РегистрироватьПоказателиСерверныхОповещений`,
- константа `СостояниеОтправкиСерверныхОповещений`,
- справочник `ВерсииРасширений`,
- справочник `ИдентификаторыОбъектовМетаданных`,
- справочник `ИдентификаторыОбъектовРасширений`,
- регистр сведений `ПараметрыРаботыПрограммы`,
- регистр сведений `ИдентификаторыОбъектовВерсийРасширений`,
- регистр сведений `ОтправленныеСерверныеОповещения`,
- регистр сведений `ПараметрыРаботыВерсийРасширений`,
- регистр сведений `ПериодическиеСерверныеОповещения`,
- регистр сведений `СвойстваРасширений`.

Банки

Подсистема **Банки** предназначена для загрузки, хранения и предоставления другим подсистемам конфигурации классификатора банков. Загрузка классификатора может производиться как автоматически с портала 1С:ИТС, так и администратором из указанного файла (при наличии подсистемы **Работа с классификаторами** библиотеки **1С:Библиотека интернет-поддержки**).

Настройка

Для использования подсистемы в конфигурации необходимо разместить в пользовательском интерфейсе ответственного за НСИ справочник `КлассификаторБанков`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Банки** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Чтение записей классификатора банков.
2.	`ДобавлениеИзменениеБанков` Ведение классификатора банков. Как правило, назначается совместно с ролью для автоматического обновление классификатора банков.
3.	`ПолучениеОбновленийКлассификаторов` (из библиотеки **1С:Библиотека интернет-поддержки**) Обновление классификатора банков РФ с портала 1С:ИТС.
4.	`ПолныеПрава`, `АдминистрированиеСистемы` (из подсистемы **Базовая функциональность**) Редактирование, удаление помеченных на удаление записей классификатора банков.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Банки» без подсистемы «Контактная информация».

Использование при разработке конфигурации

Данные классификатора банков хранятся в справочнике `КлассификаторБанков`. Данные о банках, измененные пользователем вручную, хранятся в справочниках банковских счетов. См. справочник `_ДемоБанковскиеСчета` в демонстрационной конфигурации.

Для обращения к данным банковского счета при формировании печатной формы нужно применять следующий подход, проиллюстрированный на примере документа `_ДемоСчетНаОплатуПокупателю`. Фрагмент текста запроса для выбора реквизитов банка:

```
ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанка
            ТОГДА БанковскийСчетКонтрагента.БИКБанка
        ИНАЧЕ КлассификаторБанков.Код
    КОНЕЦ КАК БИКБанк,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанка
            ТОГДА БанковскийСчетКонтрагента.НаименованиеБанка
        ИНАЧЕ КлассификаторБанков.Наименование
    КОНЕЦ КАК НаименованиеБанка,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанка
            ТОГДА БанковскийСчетКонтрагента.КоррСчетБанка
        ИНАЧЕ КлассификаторБанков.КоррСчет
    КОНЕЦ КАК КоррСчетБанка,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанка
            ТОГДА БанковскийСчетКонтрагента.ГородБанка
        ИНАЧЕ КлассификаторБанков.Город
    КОНЕЦ КАК ГородБанка,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанкаДляРасчетов
            ТОГДА БанковскийСчетКонтрагента.БИКБанкаДляРасчетов
        ИНАЧЕ КлассификаторБанковКорреспондентов.Код
    КОНЕЦ КАК БИКБанкаДляРасчетов,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанкаДляРасчетов
            ТОГДА БанковскийСчетКонтрагента.НаименованиеБанкаДляРасчетов
        ИНАЧЕ КлассификаторБанковКорреспондентов.Наименование
    КОНЕЦ КАК НаименованиеБанкаДляРасчетов,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанкаДляРасчетов
            ТОГДА БанковскийСчетКонтрагента.КоррСчетБанкаДляРасчетов
        ИНАЧЕ КлассификаторБанковКорреспондентов.КоррСчет
    КОНЕЦ КАК КоррСчетБанкаДляРасчетов,
    ВЫБОР
        КОГДА БанковскийСчетКонтрагента.РучноеИзменениеРеквизитовБанкаДляРасчетов
            ТОГДА БанковскийСчетКонтрагента.ГородБанкаДляРасчетов
        ИНАЧЕ КлассификаторБанковКорреспондентов.Город
    КОНЕЦ КАК ГородБанкаДляРасчетов,
    СчетНаОплатуПокупателю.Контрагент.НаименованиеПолное КАК ПолучательНаименованиеПолное,
    БанковскийСчетКонтрагента.НомерСчета КАК ПолучательНомерСчета
ИЗ
    Документ._ДемоСчетНаОплатуПокупателю КАК СчетНаОплатуПокупателю
        ЛЕВОЕ СОЕДИНЕНИЕ Справочник.КлассификаторБанков КАК КлассификаторБанков
        ПО СчетНаОплатуПокупателю.БанковскийСчет.Банк = КлассификаторБанков.Ссылка
        ЛЕВОЕ СОЕДИНЕНИЕ Справочник.КлассификаторБанков КАК КлассификаторБанковКорреспондентов
        ПО СчетНаОплатуПокупателю.БанковскийСчет.БанкДляРасчетов = КлассификаторБанковКорреспондентов.Ссылка
        ЛЕВОЕ СОЕДИНЕНИЕ Справочник._ДемоБанковскиеСчета КАК БанковскийСчетКонтрагента
        ПО СчетНаОплатуПокупателю.БанковскийСчет = БанковскийСчетКонтрагента
```

Пример кода по выводу реквизитов банка в табличный документ:

```
ДанныеПечати = Новый Структура;
//...
Если ПустаяСтрока(Шапка.БИКБанкаДляРасчетов) Тогда
    ДанныеПечати.Вставить("БанкПолучателяПредставление", СокрЛП(Шапка.НаименованиеБанка) + " " + СокрЛП(Шапка.ГородБанка));
    ДанныеПечати.Вставить("ПредставлениеПоставщика", СокрЛП(Шапка.ПолучательНаименованиеПолное));
    ДанныеПечати.Вставить("БИКБанкаПолучателя", СокрЛП(Шапка.БИКБанк));
    ДанныеПечати.Вставить("СчетБанкаПолучателяПредставление", СокрЛП(Шапка.КоррСчетБанка));
    ДанныеПечати.Вставить("СчетПолучателяПредставление", СокрЛП(Шапка.ПолучательНомерСчета));
Иначе
    ДанныеПечати.Вставить("БанкПолучателяПредставление", СокрЛП(Шапка.НаименованиеБанкаДляРасчетов) + " " + СокрЛП(Шапка.ГородБанкаДляРасчетов)
    ДанныеПечати.Вставить("ПредставлениеПоставщика", СокрЛП(Шапка.ПолучательНаименованиеПолное) + " р/с " + СокрЛП(Шапка.ПолучательНомерСчета) +
    ДанныеПечати.Вставить("БИКБанкаПолучателя", СокрЛП(Шапка.БИКБанкаДляРасчетов));
    ДанныеПечати.Вставить("СчетБанкаПолучателяПредставление", СокрЛП(Шапка.КоррСчетБанкаДляРасчетов));
    ДанныеПечати.Вставить("СчетПолучателяПредставление", СокрЛП(Шапка.КоррСчетБанка));
КонецЕсли;
//...
Для Каждого ИмяОбласти Из МассивОбластейМакета Цикл
    ОбластьМакета = Макет.ПолучитьОбласть(ИмяОбласти);
    //...
        ЗаполнитьЗначенияСвойств(ОбластьМакета.Параметры, ДанныеПечати);
        ТабличныйДокумент.Вывести(ОбластьМакета);
    //...
КонецЦикла;
```

Пример реализации печатной формы см. в демонстрационной конфигурации в документе `_ДемоСчетНаОплатуПокупателю`.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места, кроме следующих:

- константа `ИспользоватьАльтернативныйСерверДляЗагрузкиКлассификатораБанков`.

Бизнес-процессы и задачи

Подсистема **Бизнес-процессы и задачи** предназначена для управления жизненным циклом задач, возникающих в системе как результат выполнения бизнес-процессов или интерактивного ввода пользователями системы. Задачи могут быть адресованы исполнителю или группе исполнителей как персонально (персональная адресация), так и с использованием ролей исполнителей (ролевая адресация). В подсистему входят пять функциональных блоков: настройка ролевой адресации, создание, исполнение, контроль и автоматический мониторинг задач.

Подсистема также предоставляет базовую функциональность, облегчающую разработку произвольных бизнес-процессов в конфигурации.

Настройка

Определение состава ролей исполнителей и объектов адресации в конфигурации

Необходимость в ролевой адресации обусловлена, как правило, применением в конфигурации бизнес-процессов. Подсистема **Бизнес-процессы и задачи** предоставляет разработчикам бизнес-процессов возможности по созданию ролей исполнителей, участвующих в бизнес-процессах, а также предъявляет определенные требования к разработке самих бизнес-процессов в конфигурации.

Справочник `РолиИсполнителей` содержит «плоский» список ролей исполнителей, участвующих в бизнес-процессах. Например, **Руководитель компании**, **Менеджер продаж**, **Разработчик** и т. п. Для каждой роли в этом справочнике необходимо создать предопределенный элемент, указав значения реквизитов `Имя` и `Наименование`. Значения остальных реквизитов элементов справочника должны устанавливаться в процедуре обновления информационной базы.

С подсистемой поставляется одна предопределенная роль **Координатор выполнения задач**, которая используется для получения адресатов уведомления в функциональном блоке автоматического мониторинга задач. Например, код процедуры обновления для установки реквизитов роли **Координатор выполнения задач** выглядит так:

```
ВсеОбъектыАдресации = ПланыВидовХарактеристик.ОбъектыАдресацииЗадач.ВсеОбъектыАдресации;
РольОбъект = Справочники.РолиИсполнителей.ОтветственныйЗаКонтрольИсполнения.ПолучитьОбъект();
РольОбъект.ИспользуетсяБезОбъектовАдресации = Истина;
РольОбъект.ИспользуетсяСОбъектамиАдресации = Истина;
РольОбъект.ТипыОсновногоОбъектаАдресации = ВсеОбъектыАдресации;
РольОбъект.Записать();
```

В определенных случаях указания одной только роли исполнителя бывает недостаточно, чтобы определить список исполнителей задачи. Тогда вместе с ролью нужно еще указывать основной и/или дополнительный объект адресации. Объектом адресации может быть любой справочник конфигурации, например **Проекты** или **Подразделения**. Список возможных типов объектов адресации для ролей, определенных в справочнике `РолиИсполнителей`, содержится в плане видов характеристик (ПВХ) `ОбъектыАдресацииЗадач`. Для каждого объекта адресации в этом ПВХ необходимо создать один предопределенный элемент, указав значения реквизитов `Имя`, `Наименование` и `Тип`. Например, если необходимо, чтобы пользователь вместе с ролью **Руководитель проекта** всегда указывал проект (элемент справочника **Проекты**), то нужно сделать следующее:

- создать предопределенный элемент ПВХ `ОбъектыАдресацииЗадач`, у которого в составе типов указать ссылку на справочник **Проекты**;
- в реквизите `ТипыОсновногоОбъектаАдресации` роли **Руководитель проекта** указать ссылку на этот элемент ПВХ;
- реквизит `ИспользуетсяБезОбъектовАдресации` роли **Руководитель проекта** установить в значение `Ложь`, а `ИспользуетсяСОбъектамиАдресации` – в значение `Истина`.

Не рекомендуется создавать элементы ПВХ `ОбъектыАдресацииЗадач` с составным типом данных. Вместо этого необходимо создавать несколько предметно-ориентированных ролей, каждая из которых соответствует объекту адресации одного типа. Исключение составляет предопределенный элемент `ВсеОбъектыАдресации`, который поставляется вместе с подсистемой. Состав типов предопределенного элемента `ВсеОбъектыАдресации` должен всегда совпадать с составом типов ПВХ `ОбъектыАдресацииЗадач`.

Не следует создавать предопределенный элемент ПВХ `ОбъектыАдресацииЗадач`, когда используется только один объект адресации. Вместо этого необходимо у элемента `ВсеОбъектыАдресации` изменить наименование на представление этого объекта адресации (например, **Организация**) и указать его тип.

Добавление новых элементов ПВХ `ОбъектыАдресацииЗадач` или редактирование существующих в режиме **1С:Предприятие** недоступны.

При разработке бизнес-процессов, позволяющих организовывать взаимодействие с внешними пользователями (партнерами, респондентами и др. – подробнее см. раздел Пользователи), необходимо принять решение по поводу состава ролей, доступных для выбора внешним пользователям. Например, это могут быть **Служба техподдержки**, **Менеджер по продажам**, **Служба доставки** и т. п. Как правило, внешним пользователям недоступны имена конкретных сотрудников компании, поэтому в качестве исполнителя они могут указывать только предназначенные для них роли исполнителей. Например, код процедуры обновления для ограничения назначения роли `МенеджерПоПродажам` выглядит так:

```
РольОбъект = Справочники.РолиИсполнителей.МенеджерПоПродажам.ПолучитьОбъект();
РольОбъект.ИспользуетсяБезОбъектовАдресации = Истина;
  НазначениеРоли = РольОбъект.Назначение.Добавить();
  НазначениеРоли.ТипПользователей = Справочники.Партнеры.ПустаяСсылка();
  НазначениеРоли = РольОбъект.Назначение.Добавить();
  НазначениеРоли.ТипПользователей = Справочники.КонтактныеЛицаПартнеров.ПустаяСсылка();
РольОбъект.Записать();
```

Настройка отложенного старта для бизнес-процессов

Подсистема также позволяет создавать бизнес-процессы, старт которых может быть отложен на определенный срок. Все бизнес-процессы, которые могут быть добавлены для отложенного старта, необходимо включить в состав определяемого типа `ОтложенныйБизнесПроцесс`. В состав данного типа по умолчанию уже включен поставляемый бизнес-процесс `Задание`.

Для настройки отложенного старта бизнес-процесса необходимо разместить в форме бизнес-процесса соответствующую команду, вызывающую метод программного интерфейса `БизнесПроцессыИЗадачиКлиент.НастроитьОтложенныйСтарт`, которая и вызывает форму настройки.

Бизнес-процессы, которым настроен отложенный старт, периодически анализируются на необходимость их старта регламентным заданием `СтартОтложенныхПроцессов`.

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, то на рабочем месте администратора приложения необходимо разместить константы:

- `ИспользоватьБизнесПроцессыИЗадачи`,
- `ИспользоватьПодчиненныеБизнесПроцессы`,
- `ИзменятьЗаданияЗаднимЧислом`,
- `ИспользоватьДатуИВремяВСрокахЗадач`,
- `ИспользоватьДатуНачалаЗадач`.

См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

Для использования в конфигурации функционального блока настройки ролевой адресации необходимо:

- разместить в командном интерфейсе рабочего места администратора системы команду `РолиИИсполнителиЗадач` регистра сведений `ИсполнителиЗадач`;
- перечислить в составе типов определяемого типа `ОбъектАдресации` ссылки на справочники конфигурации, которые являются объектами адресации (состав типов команды должен совпадать с составом типов ПВХ `ОбъектыАдресацииЗадач`).

Для использования в конфигурации функционального блока создания задач необходимо:

- принять решение по поводу допустимых типов объектов метаданных, на основании которых в конфигурации будет возможно создавать задачи, буквально находясь в форме объекта. Такие объекты называются предметами задач. В качестве предметов могут выступать любые справочники, документы, планы видов характеристик и задачи;
- указать состав типов предметов задач:
  - в свойстве `Вводится на основании` бизнес-процесса `Задание`;
  - в свойстве `Тип` определяемого типа `ПредметЗадачи`;
- указать все типы бизнес-процессов в свойстве `Тип` определяемого типа `БизнесПроцесс`.

Для использования в конфигурации функционального блока исполнения задач необходимо:

- разместить в командном интерфейсе исполнителя задач команду `МоиЗадачи` задачи `ЗадачаИсполнителя` и отчеты `Задачи`, `ЗадачиИстекающиеНаДату`;
- для размещения списка своих задач на рабочем столе предназначена форма `МоиЗадачиДляРабочегоСтола` задачи `ЗадачаИсполнителя`.

Для использования в конфигурации функционального блока контроля исполнения необходимо разместить в командном интерфейсе:

- команду `МоиБизнесПроцессы` регистра сведений `СписокБизнесПроцессов`;
- команду `ВсеЗадачи` задачи `ЗадачаИсполнителя`;
- отчеты `ЗадачиИстекающиеНаДату`, `ПросроченныеЗадачи`, `Задачи`, `ЗависшиеЗадачи` и `БизнесПроцессы`.

Особые случаи внедрения подсистемы

- Внедрение подсистем «Бизнес-процессы и задачи» и «Управление доступом».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Бизнес-процессы и задачи** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ДобавлениеИзменениеРолейИсполнителей` Ведение ролей исполнителей и объектов адресации, изменение исполнителей ролей по объектам адресации, просмотр отчетов подсистемы.
2.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение использования бизнес-процессов и задач. Удаление помеченных на удаление объектов подсистемы.
3.	`ДобавлениеИзменениеЗаданий` Ввод и редактирование разрешенных бизнес-процессов **Задание**, просмотр отчетов **Список истекающих задач**, **Список просроченных задач**, **Задачи**. Роль содержит ограничения доступа, которые используют возможности подсистемы **Управление доступом**.
4.	`ИзменениеВыполнениеЗадач` Изменение и выполнение задач, направленных исполнителю. Просмотр отчетов **Список истекающих задач**, **Задачи**. Роль содержит ограничения доступа, которые используют возможности подсистемы **Управление доступом**.
5.	`ЧтениеЗаданий` Просмотр разрешенных бизнес-процессов **Задание** и разрешенных задач. Просмотр отчетов: **Список истекающих задач**, **Список просроченных задач**, **Задачи**. Роль содержит ограничения доступа, которые используют возможности подсистемы **Управление доступом**.

Дополнительно следует создать роли (или использовать подходящие роли, существующие в конфигурации) для обеспечения доступа к данным, которые не относятся к подсистеме **Бизнес-процессы и задачи**, но требуются для работы с ней.

№	Вспомогательные роли и их назначение
1.	`<ЧтениеБизнесПроцессов>` Роль или группа ролей для просмотра других бизнес-процессов конфигурации.
2.	`<ДобавлениеИзменениеБизнесПроцессов>` Роль или группа ролей для добавления и изменения других бизнес-процессов конфигурации.
3.	`<ДобавлениеИзменениеИсполнителейРолейПоОбъектамАдресации>` Изменение исполнителей ролей по разрешенным объектам адресации, просмотр отчетов подсистемы. Пример настройки прав к объектам метаданных см. в роли `_ДемоДобавлениеИзменениеИсполнителейРолейПоОбъектамАдресации` в демонстрационной конфигурации.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор.	• `ПолныеПрава` (из подсистемы **Базовая функциональность**).
2.	Ответственный за ведение списка ролей и назначение на них исполнителей.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеРолейИсполнителей`.
3.	Руководитель подразделения: • назначение исполнителей на роли в своем подразделении, • выдача заданий, • контроль выполнения задач, • просмотр отчетов по задачам подразделения.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеИсполнителейРолейПоОбъектамАдресации`, • `ДобавлениеИзменениеЗаданий`, • `ИзменениеВыполнениеЗадач`, • `ДобавлениеИзменениеБизнесПроцессов`.
4.	Сотрудник подразделения: • исполнение задач, • просмотр отчетов по своим задачам.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеЗаданий`, • `ИзменениеВыполнениеЗадач`, • `ЧтениеБизнесПроцессов`.

Использование при разработке конфигурации

Подсистема предоставляет базовую функциональность для разработки произвольных бизнес-процессов в конфигурации.

Требования к метаданным бизнес-процессов

Для унификации бизнес-процессов по общим реквизитам необходимо соблюдать следующие требования к метаданным бизнес-процессов:

- реквизит `Номер` бизнес-процесса должен быть типа `Строка`, длина – 11;
- у бизнес-процесса должны быть определены обязательные реквизиты:
  - `Автор` (`СправочникСсылка.Пользователи`) – ответственный за данный экземпляр бизнес-процесса. Например, для бизнес-процесса **Задание** автором по умолчанию является пользователь, инициировавший бизнес-процесс. Если бизнес-процесс позволяет организовать взаимодействие с внешними пользователями (партнерами, респондентами и др.), то реквизит `Автор` может быть типа `СправочникСсылка`, `ВнешниеПользователи` или составного типа: `СправочникСсылка.Пользователи` и `СправочникСсылка.ВнешниеПользователи`;
  - `ДатаЗавершения` (`Дата`) – дата фактического завершения бизнес-процесса;
  - `Наименование` (`Строка, 250, переменная длина`) – текстовое описание экземпляра бизнес-процесса. Например, в бизнес-процессе **Задание** в качестве наименования выступает текст задачи;
- в свойстве **Задачи** бизнес-процесса следует указывать задачу `ЗадачаИсполнителя`;
- в свойство **Поля блокировки данных** необходимо включить стандартный реквизит `ГлавнаяЗадача`;
- если предполагается использовать механизмы остановки и продолжения бизнес-процессов, то в бизнес-процессе следует определить реквизит `Состояние` (`ПеречислениеСсылка.СостоянияБизнесПроцессов`) и разместить на форме бизнес-процесса команды остановки и продолжения по аналогии с бизнес-процессом **Задание**;
- если предполагается использовать иерархию бизнес-процессов, то в бизнес-процессе следует определить реквизит `ГлавнаяЗадача` (`ЗадачаСсылка.ЗадачаИсполнителя`) и вывести его на форме бизнес-процесса по аналогии с бизнес-процессом **Задание**. Реквизит `ГлавнаяЗадача` необходимо включить в свойство **Поля блокировки данных**.

В модуле менеджера бизнес-процесса должны быть определены три экспортные процедуры и функции:

- `ФормаВыполненияЗадачи`,
- `ПриПеренаправленииЗадачи`,
- `ОбработкаВыполненияПоУмолчанию`.

Назначение, описание и пример их использования см. в модуле менеджера бизнес-процесса **Задание**.

Список всех бизнес-процессов, в модулях менеджеров которых определены эти процедуры и функции, необходимо перечислить в процедуре `ПриОпределенииБизнесПроцессов` общего модуля `БизнесПроцессыИЗадачиПереопределяемый`, а также включить в состав определяемых типов `БизнесПроцесс` и `БизнесПроцессОбъект`.

Если бизнес-процесс связан с некоторым объектом-основанием, в разрезе которого должен ограничиваться доступ к экземплярам бизнес-процесса, то такой бизнес-процесс должен иметь реквизит со ссылкой на объект-основание. Например, это может быть реквизит `Предмет` составного типа, как у бизнес-процесса **Задание**.

Для вывода понятных представлений бизнес-процессов в виде «Согласовать проект по Такси от 01.04.2020 (Задание)» следует также создать две подписки на события менеджеров бизнес-процессов `ОбработкаПолученияПолейПредставления` и `ОбработкаПолученияПредставления`. В качестве обработчиков подписок указать процедуры общего модуля `БизнесПроцессыИЗадачиКлиентСервер`:

- `ОбработкаПолученияПолейПредставленияБизнесПроцесса`,
- `ОбработкаПолученияПредставленияБизнесПроцесса`.

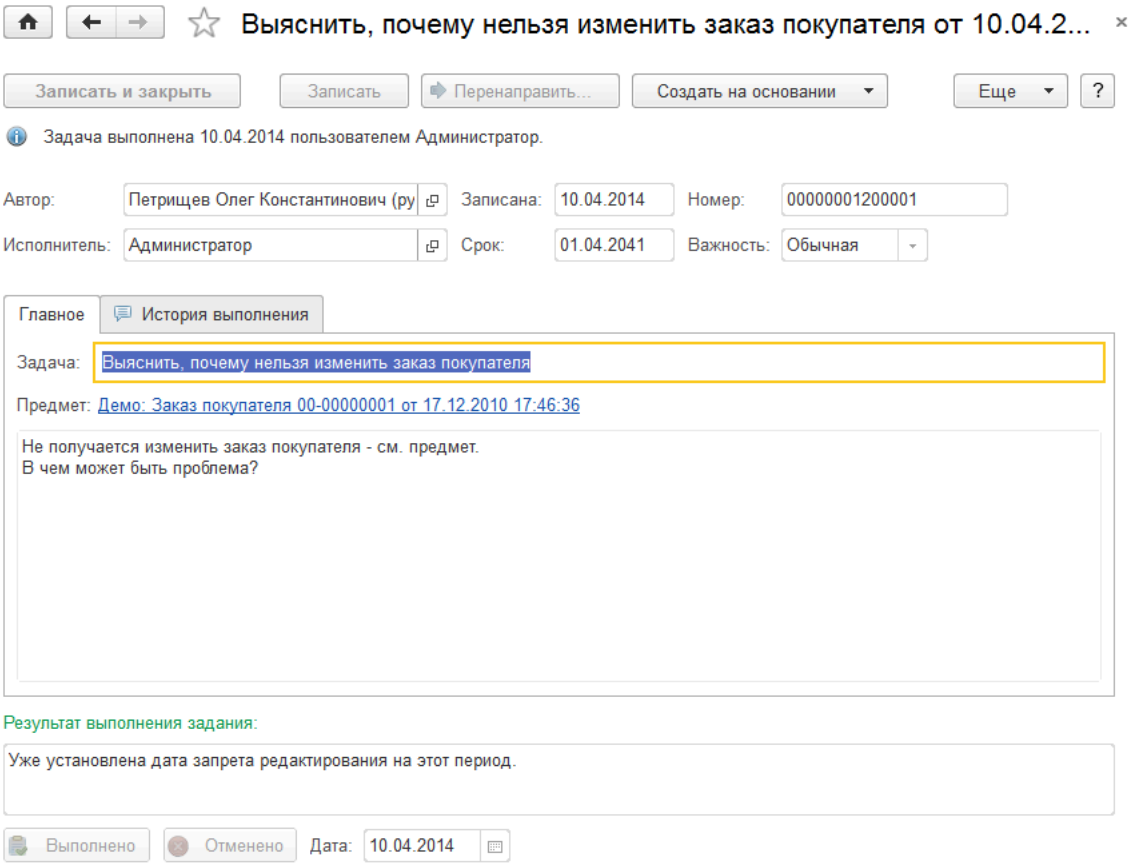
В качестве примера см. подписки на события `ПолучитьПоляПредставленияБизнесПроцесса` и `ПолучитьПредставлениеБизнесПроцесса`.

Разработка прикладных форм выполнения задач

Бизнес-процессы могут подменять форму выполнения задач по умолчанию, которая реализована в задаче `ЗадачаИсполнителя`, и использовать свою прикладную форму. Для создания удобного пользовательского интерфейса исполнителя задачи рекомендуется использовать прикладные формы, сгенерированные в точках действия карты маршрута бизнес-процесса.

Внешний вид прикладной формы выполнения задачи рекомендуется выстраивать сверху вниз по порядку:

1. Опциональное информационное поле с информацией о дате и результате выполнения задачи. Поле спрятано, если задача еще выполняется.
2. Информация об отправителе задачи (как правило, это поле с автором бизнес-процесса), общие параметры выполнения задачи, такие как **Крайний срок**, **Важность**, **Дата начала**.
3. Содержание задачи, которое может включать текстовое описание постановки задачи, ссылку на предмет задачи, а также другие реквизиты бизнес-процесса (группа **Содержание**).
4. Результат выполнения задачи, который обязательно включает поля с исполнителем и датой фактического выполнения задачи. Результат может быть многовариантным; представляться как текстовым описанием, так и специализированными элементами управления формы (флажки, переключатели, списки выбора и т. п.). В этой же группе в правом нижнем углу формы также следует размещать в виде кнопок команды для выбора варианта выполнения задачи. Например: **Выполнено**, **Отменено**, **Утверждено**, **Отработано** и т. п. Выполнение такой команды приводит к закрытию формы задачи и переходу бизнес-процесса к следующей точке маршрута.



Примеры прикладных форм можно посмотреть в базовом бизнес-процессе **Задание**.

Для подключения прикладных форм выполнения задач в модуле менеджера бизнес-процесса необходимо определить экспортную функцию-обработчик `ФормаВыполненияЗадачи`. Функция `ФормаВыполненияЗадачи` вызывается каждый раз перед открытием формы задачи. Возвращаемое значение функции – структура с ключами `ИмяФормы` и `ПараметрыФормы`, значения которых используются для открытия формы с помощью метода контекста `ОткрытьФорму`.

Пример реализации функции `ФормаВыполненияЗадачи`:

```
Функция ФормаВыполненияЗадачи(ЗадачаСсылка, ТочкаМаршрутаСсылка) Экспорт
    Результат = Новый Структура;
    Если ТочкаМаршрутаСсылка = БизнесПроцессы.Задание.ТочкиМаршрута.Выполнить Тогда
        Результат.Вставить("ПараметрыФормы", Новый Структура("Ключ", ЗадачаСсылка));
        Результат.Вставить("ИмяФормы", "БизнесПроцесс.Поручение.Форма.Выполнить");
    КонецЕсли;
    Возврат Результат;
КонецФункции
```

Дополнительные рекомендации по разработке метаданных и прикладных форм выполнения задач:

- Рекомендуется давать имена формам выполнения задач по определенному шаблону: `Действие<ИмяТочкиМаршрута>`. Например, `ДействиеВыполнить`.
- Прикладная форма выполнения задачи обязательно должна содержать основной реквизит `Объект` типа `ЗадачаОбъект.ЗадачаИсполнителя`, а также может содержать произвольный набор дополнительных реквизитов, среди которых, как правило, есть реквизиты бизнес-процесса и объектов, связанных с процессом выполнения задачи.
- Если в форме присутствуют реквизиты бизнес-процесса (или других связанных объектов), то чтение и запись этих реквизитов рекомендуется выполнять в привилегированном режиме. Это позволит работать с формой задачи тем пользователям, которые при своей повседневной работе с системой не имеют прав на чтение или запись реквизитов бизнес-процесса, но, тем не менее, косвенно должны иметь к ним доступ из формы задачи.

Пример:

```
&НаСервере
Процедура ПрочитатьРеквизитыБизнесПроцесса()

    ЗадачаОбъект = РеквизитФормыВЗначение("Объект");

    УстановитьПривилегированныйРежим(Истина);
    ЗаданиеОбъект = ЗадачаОбъект.БизнесПроцесс.ПолучитьОбъект();
    ЗаданиеВыполнено = ЗаданиеОбъект.Выполнено;
    ЗаданиеРезультатВыполнения = ЗаданиеОбъект.РезультатВыполнения;
    ЗаданиеСодержание = ЗаданиеОбъект.Содержание;

КонецПроцедуры
&НаСервере
Процедура ЗаписатьРеквизитыБизнесПроцесса(ЗадачаОбъект)

    УстановитьПривилегированныйРежим(Истина);
    ЗаданиеОбъект = ЗадачаОбъект.БизнесПроцесс.ПолучитьОбъект();
    ЗаблокироватьДанныеДляРедактирования(ЗаданиеОбъект.Ссылка);
    ЗаданиеОбъект.Выполнено = ЗаданиеВыполнено;
    ЗаданиеОбъект.Записать();

КонецПроцедуры
```

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места. Исключение из этого правила:

- константу `ДатаУведомленияОНовыхЗадачах` не следует включать в планы обмена РИБ, если требуется независимо уведомлять исполнителей о новых задачах в узлах РИБ.

Валюты

Подсистема **Валюты** предназначена для загрузки, хранения и предоставления другим подсистемам конфигурации списка валют и их курсов. Загрузка классификатора валют может производиться как автоматически с портала 1С:ИТС, так и администратором из указанного файла (при наличии подсистемы **Работа с классификаторами** библиотеки **1С:Библиотека интернет-поддержки**). Загрузка курсов валют может производиться как автоматически (при помощи регламентного задания), так и вручную (через пользовательский интерфейс справочника **Валюты**).

Настройка

Для использования подсистемы в конфигурации необходимо разместить в командном интерфейсе методиста-настройщика справочник **Валюты**.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Валюты** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ЧтениеКурсовВалют` Просмотр валют и курсов.
2.	`ДобавлениеИзменениеКурсовВалют` Редактирование списка валют и курсов. Как правило, назначается совместно с ролью для автоматической загрузки курсов валют.
3.	• `ПолучениеОбновленийКлассификаторов` (из библиотеки **1С:Библиотека интернет-поддержки**) Обновление классификатора банков РФ с портала 1С:ИТС.
4.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Удаление помеченных на удаление объектов подсистемы.

Использование при разработке конфигурации

Список валют и их курсы хранятся в справочнике **Валюты** и регистре сведений **Курсы валют** соответственно.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные рекомендуется включить в планы обмена распределенной ИБ (РИБ) и автономного рабочего места. Исключение составляет:

- константа `ИспользоватьАльтернативныйСерверДляЗагрузкиКурсовВалют`.

Варианты отчетов

Подсистема **Варианты отчетов** отображает все отчеты раздела на панели отчетов, предоставляет совместный доступ к пользовательским вариантам отчетов, а также универсальные формы для формирования и настройки произвольных отчетов конфигурации.

Варианты отчетов, созданные разработчиком, называются предопределенными, а сохраненные пользователем в режиме **1С:Предприятие** – пользовательскими. Предопределенные варианты отчетов могут быть контекстными или неконтекстными:

- контекстные варианты отчетов могут использоваться только в контексте работы с каким-либо объектом приложения (например, отчет по сотруднику, подразделению и т. п.). Как правило, такие варианты отчетов открываются из формы объекта и списков;
- обычные варианты отчетов могут одинаково открываться из любого места приложения (например, из панели отчетов), не требуя передачи контекста. Как правило, такие варианты отчетов открываются из командного интерфейса.

Начальные настройки предопределенных вариантов отчетов зачитываются из метаданных, и их можно изменить на этапе внедрения подсистемы. Для того чтобы изменения в метаданных конфигурации (добавление, изменение отчетов, вариантов отчетов и т.п.) сразу вступили в силу в процессе разработки и отладки, необходимо обновить данные справочника ВариантыОтчетов.

Размещение варианта отчета на панели отчетов можно изменить при сохранении или из его карточки. Настройки размещения выводятся в виде дерева подсистем с флажком использования и важностью.

В панели отчетов раздела каждый пользователь может изменить настройки видимости и быстрого доступа к вариантам отчетов под себя, переключившись в режим настройки. Также в панели отчетов предусмотрен поиск по свойствам отчета: наименованию, описанию, наименованиям подсистем (разделов и групп командного интерфейса), полей, отборов и пользовательских настроек.

Настройка

Настройка подсистемы состоит из следующих этапов:

1. Подключить хранилище вариантов отчетов.
2. Подключить форму отчета.
3. Подключить подсистему к разделам командного интерфейса.
4. Установить настройки вариантов отчетов.
5. Подключить контекстные отчеты.
6. Установить настройки формы отчета.
7. Обновить данные справочника `ВариантыОтчетов`.

Подключить хранилище вариантов отчетов

Необходимо принять решение по поводу варианта внедрения хранилища подсистемы:

1. Если конфигурация ранее не выпускалась (т. е. пользователи не сохраняли в своих базах никаких вариантов отчетов), то следует выбрать один из двух способов внедрения:
   - **Полный**, если к подсистеме необходимо подключить все отчеты конфигурации, а также все внешние (дополнительные) отчеты. Для этого необходимо установить хранилище настроек `ХранилищеВариантовОтчетов` в свойстве конфигурации **Хранилище вариантов отчетов**;
   - **Частичный**, если необходимо подключить подсистему выборочно только для некоторой части отчетов. Для этого необходимо установить хранилище настроек `ХранилищеВариантовОтчетов` в свойстве **Хранилище вариантов отчетов** каждого отчета, который нужно подключить к хранилищу подсистемы.
2. Если конфигурация уже выпускалась, т. е. существует вероятность того, что пользователи сохраняли в своих базах варианты отчетов, то следует использовать только **Частичный способ внедрения**. В этом случае ранее сохраненные пользователями варианты отчетов будут перенесены в хранилище подсистемы автоматически.
   В противном случае, если в такой ситуации был выбран сценарий внедрения **Полный**, подсистема не сможет автоматически перенести ранее сохраненные пользователями варианты отчетов. В этом случае необходимо:
   - включить в сопроводительную документацию к конфигурации инструкцию (или донести эту информацию другими способами) для администратора (пользователя, который выполняет обновление конфигурации), что он должен запустить обработку перехода перед обновлением;
   - разработать и включить в состав дистрибутива обработку перехода. Пример подобной обработки входит в состав демонстрационной конфигурации – `ПереносВариантовОтчетов.epf`.

Подключить форму отчета

Необходимо принять решение по поводу варианта внедрения формы отчета:

- **Полный**, если к подсистеме необходимо подключить все отчеты без основной формы. Для этого необходимо указать в свойствах конфигурации общие формы, поставляемые подсистемой:
  - основная форма отчета – `ФормаОтчета`;
  - основная форма настроек отчета – `ФормаНастроекОтчета`;
  - основная форма варианта отчета – `ФормаВариантаОтчета`.
- **Частичный**, если необходимо подключить подсистему выборочно только для части отчетов без основной формы. Для этого необходимо указать в свойствах каждого отчета общие формы, поставляемые подсистемой:
  - основная форма – `ФормаОтчета`;
  - основная форма настроек – `ФормаНастроекОтчета`;
  - основная форма варианта – `ФормаВариантаОтчета`.

> **Внимание!**
>
> Не поддерживается ситуация, когда отчет подключен только к некоторым из указанных форм. В частности, в случае «Полного» варианта внедрения, если для отчета на СКД определена собственная основная форма, то для него также следует определить собственную форму настроек (либо в свойствах этого отчета указать общую форму `ВспомогательнаяФормаНастроекОтчета`). Это требование также распространяется на дополнительные и внешние отчеты, использующиеся в конфигурации с «Полным» вариантом внедрения формы отчета.

Подключить подсистему к разделам командного интерфейса

Необходимо определить разделы командного интерфейса, в которых должны быть размещены панели отчетов. Рекомендуется выбрать все разделы, для которых предусмотрены отчеты.

- Для каждого раздела необходимо создать отдельную общую команду для открытия панели отчетов.
  - Имя команды рекомендуется задавать по шаблону `ПанельОтчетов<ИмяРаздела>`, например `ПанельОтчетовПродажи`.
  - Синоним команды рекомендуется определять по шаблону **Отчеты по <ПредставлениеРаздела>**, например **Отчеты по продажам**.
  - В обработчике команды необходимо вставить вызов по шаблону:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполнения)
    ВариантыОтчетовКлиент.ПоказатьПанельОтчетов("<ИмяРаздела>", ПараметрыВыполнения);
КонецПроцедуры
```

  Для того чтобы в первом параметре **<ИмяРаздела>** указать начальную страницу, следует использовать функцию `ВариантыОтчетовКлиентСервер.ИдентификаторНачальнойСтраницы`:

```
ВариантыОтчетовКлиент.ПоказатьПанельОтчетов(ВариантыОтчетовКлиентСервер.ИдентификаторНачальнойСтраницы(), ПараметрыВыполнения);
```

- Выбранные разделы необходимо перечислить в процедуре `ОпределитьРазделыСВариантамиОтчетов` общего модуля `ВариантыОтчетовПереопределяемый`. В представлениях разделов (используются в качестве заголовков панелей отчетов) рекомендуется указывать синонимы команд, созданных на предыдущем шаге.
- Все отчеты выбранных разделов рекомендуется скрыть из панели действий командного интерфейса этих разделов.

Установить настройки вариантов отчетов

Настройки вариантов отчетов задаются в процедуре `НастроитьВариантыОтчетов` общего модуля `ВариантыОтчетовПереопределяемый`. Настройки можно менять непосредственно в переопределяемом модуле (удобно в небольших конфигурациях) или в модуле менеджера отчета (удобно при совместной разработке и больше соответствует библиотечному подходу). Для этого в процедуре `НастроитьВариантыОтчетов` общего модуля `ВариантыОтчетовПереопределяемый` следует разместить вызов модуля менеджера отчета по шаблону:

```
ВариантыОтчетов.НастроитьОтчетВМодулеМенеджера(Настройки, Метаданные.Отчеты.ИмяОтчета);
```

А в модуле менеджера отчета вставить процедуру по шаблону:

```
#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда
#Область ПрограммныйИнтерфейс
#Область ДляВызоваИзДругихПодсистем
// Параметры:
//   Настройки - см. ВариантыОтчетовПереопределяемый.НастроитьВариантыОтчетов.Настройки.
//   НастройкиОтчета - см. ВариантыОтчетов.ОписаниеОтчета.
//
Процедура НастроитьВариантыОтчета(Настройки, НастройкиОтчета) Экспорт

КонецПроцедуры
#КонецОбласти
#КонецОбласти
#КонецЕсли
```

Объекты `НастройкиОтчета` и `НастройкиВарианта` содержат следующие свойства, которые можно менять:

- `Включен` – если установить в `Ложь`, то вариант отчета не регистрируется в справочнике и, как следствие, не выводится в панелях отчетов и других формах подсистемы (всегда скрыт от пользователей). При этом вариант отчета остается в конфигурации, и его по-прежнему можно открывать при программном открытии формы отчета. Например, может потребоваться отключение контекстных вариантов отчетов.
- `ВидимостьПоУмолчанию` – если установить в `Ложь`, то вариант отчета по умолчанию будет скрыт на панелях отчетов. Скрытый вариант отчета можно вывести на панель отчетов администратору – сразу для всех пользователей (через форму элемента), пользователю – на свою панель отчетов (через режим настройки). Например, может потребоваться скрывать менее частотные варианты отчетов.
- `Описание` – служит подсказкой на панели отчетов и содержит более подробное описание варианта отчета. Свойство рекомендуется заполнять, поскольку участвует в поиске отчета.
- `Размещение` – служит для привязки вариантов отчетов к подсистемам конфигурации. Подсистемы первого уровня считаются разделами, второго и выше – группами. Если вариант включен в раздел, то на панели отчетов он будет выведен без группы. При необходимости для каждой подсистемы можно указать `Важность`:
  - `Важный` – вариант отчета будет выделен жирным шрифтом и размещен в начале группы.
  - `СмТакже` – вариант отчета будет выведен внизу панели отчетов, в группе **См. также**.

- `ФункциональныеОпции` – служит для скрытия вариантов отчетов по функциональным опциям. Содержит массив строк из имен функциональных опций конфигурации. Вариант отчета считается включенным в том случае, если включена одна из функциональных опций отчета (если они указаны) и одна из функциональных опций варианта отчета (если они указаны).
- `НастройкиДляПоиска` – дополнительные сведения для поиска этого варианта отчета. Является структурой со следующими свойствами:
  - `НаименованияПолей` – представления полей варианта отчета;
  - `НаименованияПараметровИОтборов` – представления настроек варианта отчета;
  - `КлючевыеСлова` – дополнительная терминология (в т. ч. специализированная или устаревшая).
- Разделитель терминов: `Символы.ПС`.
- `ОпределитьНастройкиФормы` – включает переопределение настроек формы отчета. Внимание: свойство доступно только для объекта `НастройкиОтчета`. Если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ОпределитьНастройкиФормы` по шаблону:

```
#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда
#Область ПрограммныйИнтерфейс
#Область ДляВызоваИзДругихПодсистем
// Параметры:
//   Форма - ФормаКлиентскогоПриложения, Неопределено
//   КлючВарианта - Строка, Неопределено
//   Настройки - см. ОтчетыКлиентСервер.НастройкиОтчетаПоУмолчанию
//
Процедура ОпределитьНастройкиФормы(Форма, КлючВарианта, Настройки) Экспорт

КонецПроцедуры
#КонецОбласти
#КонецОбласти
#КонецЕсли
```

Подробнее про изменение настроек формы отчета см. раздел Изменить настройки формы отчета.

Подключить контекстные отчеты к объектам конфигурации

Необходимо принять решение, для каких справочников, документов и других объектов конфигурации разрабатываются контекстные отчеты и в каких их формах требуется выводить подменю с контекстными отчетами. Например, команда открытия отчета о движениях документа может быть предусмотрена во всех документах, формирующих движения в регистры.

При этом стоит учитывать сценарии подключения команд при помощи расширений конфигурации и дополнительных отчетов. Т. е. если для определенного объекта конфигурации не планируется разрабатывать контекстные отчеты в составе конфигурации, то все равно рекомендуется заблаговременно подключать команды отчетов для этого объекта.

Для подключения контекстных отчетов к объектам конфигурации следует:

- Объекты метаданных, для которых требуется выводить команды отчетов, перечислить в процедуре `ОпределитьОбъектыСКомандамиОтчетов` модуля `ВариантыОтчетовПереопределяемый`.
- В модуле менеджера каждого из этих объектов в области `ПрограммныйИнтерфейс` определить пустую процедуру `ДобавитьКомандыОтчетов` по шаблону:

```
#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда
#Область ПрограммныйИнтерфейс
#Область ДляВызоваИзДругихПодсистем
// Параметры:
//  КомандыОтчетов - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.КомандыОтчетов
//  Параметры - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.Параметры
//
Процедура ДобавитьКомандыОтчетов(КомандыОтчетов, Параметры) Экспорт

КонецПроцедуры
#КонецОбласти
#КонецОбласти
#КонецЕсли
```

- В формах объектов (списков и т. п.) метаданных, в которых требуется выводить подменю **Отчеты**, встроить подсистему **Подключаемые команды**. В командную панель, для целей оптимизации производительности, рекомендуется добавить подменю для вывода команд отчетов:
  - Имя: `ПодменюОтчеты`,
  - Заголовок: `Отчеты`,
  - Вид: `Подменю`,
  - Отображение: `Картинка`.
  - Картинка: `Отчет` (стандартная картинка).

Для вывода в подменю **Отчеты** большого количества команд (более 10) рекомендуется добавить вложенные группы кнопок с суффиксами `Важное`, `Обычное` и `СмТакже`. Например: `ПодменюОтчетыВажное`, `ПодменюОтчетыОбычное` и `ПодменюОтчетыСмТакже`. Тогда разработчики команд контекстных отчетов смогут указывать суффиксы этих групп в свойстве `Важность` своих команд.

Подробнее о разработке команд контекстных отчетов см. в разделе ниже.

Установить настройки формы отчета

Настройки формы отчета задаются в процедуре `ОпределитьНастройкиФормы` модуля объекта отчета, в которой можно задать значения следующим параметрам:

- `ФормироватьСразу` – если установить в значение `Истина`, то отчет будет формироваться после открытия, после выбора пользовательских настроек и после выбора другого варианта отчета. По умолчанию – `Ложь`.
- `ВыводитьСуммуВыделенныхЯчеек` – если установить в значение `Ложь`, то в отчете не будет выводиться поле с автоматически рассчитываемой суммой выделенных ячеек. По умолчанию – `Истина`.
- `РазрешеноИзменятьСтруктуру` – если установить в значение `Ложь`, то в настройках отчета будет скрыта вкладка `Структура`. По умолчанию – `Истина`. По умолчанию вкладка `Структура` показывается для отчетов на СКД – в расширенном режиме, а также в простом режиме, если в пользовательские настройки выведены флажки использования группировок.
- `РазрешеноИзменятьВарианты` – если установить в значение `Ложь`, то в форме отчета и настроек скрываются кнопки изменения вариантов этого отчета. Если у текущего пользователя нет прав `СохранениеДанныхПользователя` и `Добавление` справочника `ВариантыОтчетов`, то принудительно устанавливается в `Ложь`. По умолчанию – `Истина`.
- `Печать` – `Структура` – параметры печати табличного документа (могут переопределяться пользователем):
  - `ПолеСверху` – `Число` – отступ сверху при печати (в миллиметрах);
  - `ПолеСлева` – `Число` – отступ слева при печати (в миллиметрах);
  - `ПолеСнизу` – `Число` – отступ снизу при печати (в миллиметрах);
  - `ПолеСправа` – `Число` – отступ справа при печати (в миллиметрах);
  - `ОриентацияСтраницы` – `ОриентацияСтраницы` – `Портрет` или `Ландшафт`;
  - `АвтоМасштаб` – `Булево` – автоматически подгонять масштаб под размер страницы;
  - `МасштабПечати` – `Число` – масштаб изображения (в процентах).
- `События` – `Структура` – события, для которых определены обработчики в модуле объекта отчета (обработчики событий рекомендуется определять в области `ПрограммныйИнтерфейс`, после процедуры `ОпределитьНастройкиФормы`):
  - `ПриСозданииНаСервере` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПриСозданииНаСервере` по шаблону:

    ```
    // См. ОтчетыПереопределяемый.ПриСозданииНаСервере.
    Процедура ПриСозданииНаСервере(Форма, Отказ, СтандартнаяОбработка) Экспорт

    КонецПроцедуры
    ```

    С ее помощью можно вывести дополнительные команды с помощью процедуры `ОтчетыСервер.ВывестиКоманду` или выполнить другие подготовительные действия в форме отчета. Пример добавления команды:

    ```
    Команда = Форма.Команды.Добавить("<ИмяКоманды>");
    Команда.Действие  = "Подключаемый_Команда";
    Команда.Заголовок = НСтр("ru = '<Представление команды...>'");
    ОтчетыСервер.ВывестиКоманду(Форма, Команда, "<ВидГруппы>");
    ```

    При этом обработчик команды нужно реализовать в процедуре `ОтчетыКлиентПереопределяемый.ОбработчикКоманды`.

  - `ПередЗагрузкойНастроекВКомпоновщик` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПередЗагрузкойНастроекВКомпоновщик` по шаблону:

    ```
    // Параметры:
    //   Контекст - Произвольный
    //   КлючСхемы - Строка
    //   КлючВарианта - Строка, Неопределено
    //   НовыеНастройкиКД - НастройкиКомпоновкиДанных, Неопределено
    //   НовыеПользовательскиеНастройкиКД - ПользовательскиеНастройкиКомпоновкиДанных, Неопределено
    //
    Процедура ПередЗагрузкойНастроекВКомпоновщик(Контекст, КлючСхемы, КлючВарианта, НовыеНастройкиКД, НовыеПользовательскиеНастройкиКД) Эк

    КонецПроцедуры
    ```

    Вызывается перед загрузкой новых настроек, используется для изменения СКД.

    Пример № 1. Компоновщик отчета инициализируется на основании схемы из общих макетов:

    ```
    Если КлючСхемы <> "1" Тогда
        КлючСхемы = "1";
        СхемаКД = ПолучитьОбщийМакет("МояОбщаяСхемаКомпоновки");
        ОтчетыСервер.ПодключитьСхему(ЭтотОбъект, Контекст, СхемаКД, КлючСхемы);
    КонецЕсли;
    ```

    Пример № 2. Схема зависит от значения параметра, выведенного в пользовательские настройки отчета:

```
Если ТипЗнч(НовыеПользовательскиеНастройкиКД) = Тип("ПользовательскиеНастройкиКомпоновкиДанных") Тогда
    ПолноеИмяОбъектаМетаданных = "";
    Для Каждого ЭлементКД Из НовыеПользовательскиеНастройкиКД.Элементы Цикл
        Если ТипЗнч(ЭлементКД) = Тип("ЗначениеПараметраНастроекКомпоновкиДанных") Тогда
            ИмяПараметра = Строка(ЭлементКД.Параметр);
            Если ИмяПараметра = "ОбъектМетаданных" Тогда
                ПолноеИмяОбъектаМетаданных = ЭлементКД.Значение;
            КонецЕсли;
        КонецЕсли;
    КонецЦикла;
    Если КлючСхемы <> ПолноеИмяОбъектаМетаданных Тогда
        КлючСхемы = ПолноеИмяОбъектаМетаданных;
        СхемаКД = Новый СхемаКомпоновкиДанных;
        // Наполнение схемы...
        ОтчетыСервер.ПодключитьСхему(ЭтотОбъект, Контекст, СхемаКД, КлючСхемы);
    КонецЕсли;
КонецЕсли;
```

- `ПослеЗагрузкиНастроекВКомпоновщик` если Истина, то в модуле объекта отчета следует определить процедуру `ПослеЗагрузкиНастроекВКомпоновщик` по шаблону:

```
//  Параметры:
//    ДополнительныеПараметры - Структура
//
Процедура ПослеЗагрузкиНастроекВКомпоновщик(ДополнительныеПараметры) Экспорт

КонецПроцедуры
```

Вызывается после загрузки всех настроек в компоновщик. Используется для уточнения, например, параметров выбора, зависимых от значений загруженных фиксированных отборов.

- `ПередЗагрузкойВариантаНаСервере` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПередЗагрузкойВариантаНаСервере` по шаблону:

```
// См. ОтчетыПереопределяемый.ПередЗагрузкойВариантаНаСервере.
Процедура ПередЗагрузкойВариантаНаСервере(Форма, НовыеНастройкиКД) Экспорт

КонецПроцедуры
```

Вызывается в обработчике одноименного события формы отчета и формы настройки отчета. Подробнее см. "Расширение управляемой формы для отчета.ПередЗагрузкойВариантаНаСервере" в синтакс-помощнике.

- `ПриЗагрузкеВариантаНаСервере` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПриЗагрузкеВариантаНаСервере` по шаблону:

```
// Параметры:
//   Форма - ФормаКлиентскогоПриложения
//         - РасширениеУправляемойФормыДляОтчета:
//     * Отчет - ДанныеФормыСтруктура
//             - ОтчетОбъект
//   НовыеНастройкиКД - НастройкиКомпоновкиДанных
//
Процедура ПриЗагрузкеВариантаНаСервере(Форма, НовыеНастройкиКД) Экспорт

КонецПроцедуры
```

Вызывается в обработчике одноименного события формы отчета после выполнения кода формы. Подробнее см. "Расширение управляемой формы для отчета.ПриЗагрузкеВариантаНаСервере" в синтакс-помощнике.

- `ПриЗагрузкеПользовательскихНастроекНаСервере` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПриЗагрузкеПользовательскихНастроекНаСервере` по шаблону:

```
// Параметры:
//   Форма - ФормаКлиентскогоПриложения
//   НовыеПользовательскиеНастройкиКД - ПользовательскиеНастройкиКомпоновкиДанных
//
Процедура ПриЗагрузкеПользовательскихНастроекНаСервере(Форма, НовыеПользовательскиеНастройкиКД) Экспорт

КонецПроцедуры
```

Вызывается в обработчике одноименного события формы отчета после выполнения кода формы. Подробнее см. "Расширение управляемой формы для отчета.ПриЗагрузкеПользовательскихНастроекНаСервере" в синтакс-помощнике.

- `ПередЗаполнениемПанелиБыстрыхНастроек` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПередЗаполнениемПанелиБыстрыхНастроек` по шаблону:

```
// Параметры:
//   Форма - ФормаКлиентскогоПриложения
//   ПараметрыЗаполнения - Структура
//
Процедура ПередЗаполнениемПанелиБыстрыхНастроек(Форма, ПараметрыЗаполнения) Экспорт

КонецПроцедуры
```

Вызывается до перезаполнения панели настроек формы отчета.

- `ПослеЗаполненияПанелиБыстрыхНастроек` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПослеЗаполненияПанелиБыстрыхНастроек` по шаблону:

```
// Параметры:
//   Форма - ФормаКлиентскогоПриложения
//   ПараметрыЗаполнения - Структура
//
Процедура ПослеЗаполненияПанелиБыстрыхНастроек(Форма, ПараметрыЗаполнения) Экспорт

КонецПроцедуры
```

Вызывается после перезаполнения панели настроек формы отчета.

- `ПриОпределенииОсновныхПолей` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПриОпределенииОсновныхПолей` по шаблону:

```
// См. ОтчетыПереопределяемый.ПриОпределенииОсновныхПолей.
Процедура ПриОпределенииОсновныхПолей(Форма, ОсновныеПоля) Экспорт

КонецПроцедуры
```

Позволяет задать список часто используемых полей, которые будут выводиться в подменю для команд контекстного меню "Вставить поле слева", "Вставить группировку ниже" и т.п.

- `ПередФормированиемОтчета` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПередФормированиемОтчета` по шаблону:

```
// Параметры:
//   ФормаОтчета - ФормаКлиентскогоПриложения
//   ДополнительныеПараметры - Структура:
//     * ТекстПредупреждения - Строка
//     * ИмяПараметраХраненияОтказаОтПредупреждения - Строка
//
Процедура ПередФормированиемОтчета(Форма, ДополнительныеПараметры) Экспорт

КонецПроцедуры
```

Позволяет перед формированием отчета показать вопрос с текстом `ТекстПредупреждения` и кнопками **Продолжить** и **Отмена**. Например, предупредить о том, что отчет будет очень долго формироваться и потребует много памяти и порекомендовать более узкую настройку отборов. Если также задать свойство `ИмяПараметраХраненияОтказаОтПредупреждения`, то в окне вопроса появится опция больше его не показывать.

- `ПриОпределенииПараметровВыбора` – если установить в значение `Истина`, то в модуле объекта отчета следует определить процедуру `ПриОпределенииПараметровВыбора` по шаблону:

```
// См. ОтчетыПереопределяемый.ПриОпределенииПараметровВыбора.
Процедура ПриОпределенииПараметровВыбора(Форма, СвойстваНастройки) Экспорт

КонецПроцедуры
```

Вызывается в форме отчета и в форме настройки отчета перед выводом настройки для указания дополнительных параметров выбора. Это устаревшее событие, вместо него рекомендуется применять событие `ПослеЗагрузкиНастроекВКомпоновщик`.

Для глобального переопределения поведения универсальной формы отчета часть серверных событий представлены в модуле `ОтчетыПереопределяемый`.

Процедуры для переопределения поведения клиентских событий, возникающих в универсальной форме отчета, представлены в модуле `ОтчетыКлиентПереопределяемый`.

В области `ПрограммныйИнтерфейс` модулей `ОтчетыКлиент` и `ОтчетыСервер` представлены вспомогательные процедуры, которые можно вызывать из обработчиков событий формы отчета и при формировании отчета.

Обновить данные справочника «ВариантыОтчетов»

> **Внимание!**
>
> Для того чтобы изменения в метаданных конфигурации (добавление, изменение отчетов, вариантов отчетов и т. п.) вступили в силу, необходимо обновить данные справочника `ВариантыОтчетов`. В большинстве случаев он обновляется автоматически при каждом изменении номера версии конфигурации – в процессе обновления или первоначального заполнения информационной базы. При ведении разработки и отладки конфигурации его рекомендуется обновлять вручную, подробнее см. раздел Обновление вспомогательных данных во время разработки.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Варианты отчетов» без подсистемы «Управление доступом».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы нужно использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Администрирование вариантов отчетов.
2.	`ДобавлениеИзменениеВариантовОтчетов` Администрирование вариантов отчетов. Если у пользователя нет права `СохранениеДанныхПользователя`, то отключаются все возможности по сохранению настроек подсистемы.
3.	`ДобавлениеИзменениеЛичныхВариантовОтчетов` Добавление и изменение своих вариантов отчетов (доступных только автору). Просмотр общих вариантов отчетов (доступных всем пользователям). Если у пользователя нет права `СохранениеДанныхПользователя`, то отключаются все возможности по сохранению настроек подсистемы.
4.	`ЧтениеВариантовОтчетов` Аудит вариантов отчетов. Просмотр любых вариантов отчетов без права изменения.
5.	• `БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**) Просмотр общих вариантов отчетов (доступных всем пользователям).
6.	• `ИспользованиеУниверсальногоОтчета` • Доступ к отчету **Универсальный отчет**.
7.	`ДобавлениеИзменениеСнимковОтчетов`. Сохранение снимков отчетов и их просмотр, в том числе в мобильном клиенте в автономном режиме.

Использование при разработке конфигурации

Для того чтобы изменения в метаданных конфигурации (добавление, изменение отчетов, вариантов отчетов и т. п.) вступили в силу, необходимо обновить данные справочника «ВариантыОтчетов». В редких случаях для этого может потребоваться разработать свой обработчик обновления в прикладной конфигурации.

При необходимости перезаполнения предопределенных вариантов отчетов можно воспользоваться процедурой `Обновить` общего модуля `ВариантыОтчетов`. Например, это может потребоваться при изменении настроек приложения, если некоторые варианты отчетов отключаются по условиям в зависимости от настроек приложения.

При необходимости сбросить пользовательские настройки отчетов можно воспользоваться процедурой `СброситьПользовательскиеНастройки` общего модуля `ВариантыОтчетов`. Например, это может потребоваться, чтобы «пробросить» изменения в настройках вариантов отчетов пользователям этого варианта.

В случае если у варианта отчета был изменен ключ, его следует указать в процедуре `ЗарегистрироватьИзмененияКлючейВариантовОтчетов` общего модуля `ВариантыОтчетовПереопределяемый`.

Описание таблиц, используемых при компоновке отчета

Некоторым механизмам требуется информация о том, какие таблицы используются при компоновке того или иного отчета. Например, для того чтобы

- выводить предупреждение, что отчет может содержать некорректные данные, так как не завершен переход на новую версию приложения;
- выводить информацию о правах отчетов в отчете **Анализ прав доступа** .

В большинстве случаев эти сведения подсистема **Варианты отчетов** извлекает автоматически из набора данных типа **Запрос** схемы компоновки данных (СКД). Однако в случае использования набора данных типа **Объект**, информация об используемых таблицах в СКД отсутствует. В этом случае необходимо в модуле объекта отчета разместить обработчик события `ПриОпределенииИспользуемыхТаблиц` и перечислить в нем используемые таблицы отчета:

```
// Задать настройки формы отчета.
//
// Параметры:
//   Форма - ФормаКлиентскогоПриложения
//         - Неопределено
//   КлючВарианта - Строка
//                - Неопределено
//   Настройки - см. ОтчетыКлиентСервер.НастройкиОтчетаПоУмолчанию
//
Процедура ОпределитьНастройкиФормы(Форма, КлючВарианта, Настройки) Экспорт
  Настройки.События.ПриОпределенииИспользуемыхТаблиц = Истина;
КонецПроцедуры

// Параметры:
//   КлючВарианта - Строка
//                - Неопределено -
//       Имя предопределенного или уникального идентификатор пользовательского варианта отчета.
//       Неопределено когда вызов для варианта расшифровки или без контекста.
//   ИспользуемыеТаблицы - Массив Из Строка
//
Процедура ПриОпределенииИспользуемыхТаблиц(КлючВарианта, ИспользуемыеТаблицы) Экспорт
  ИспользуемыеТаблицы.Добавить(Метаданные.Документы._ДемоЗаказПокупателя.ПолноеИмя());
КонецПроцедуры
```

При этом в параметр `КлючВарианта` передается имя предопределенного или уникальный идентификатор пользовательского варианта отчета, или `Неопределено`, когда вызов для варианта расшифровки или без контекста.

Разработка команд контекстных отчетов

Для разработки контекстных отчетов в конфигурации предварительно должна быть выполнена настройка тех объектов (справочников, документов и т. п.), к которым эти отчеты подключаются – см. раздел Подключить контекстные отчеты к объектам конфигурации.

Команды отчетов, выводимые в конкретных справочниках, документах и других объектах конфигурации, задаются в процедуре `ДобавитьКомандыОтчетов` модуля менеджера этого объекта метаданных. Например:

```
// Определяет список команд отчетов.
//
// Параметры:
//   КомандыОтчетов - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.КомандыОтчетов
//   Параметры - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.Параметры
//
Процедура ДобавитьКомандыОтчетов(КомандыОтчетов, Параметры) Экспорт
    Если ПравоДоступа("Просмотр", Метаданные.Отчеты.МестаИспользованияСсылок) Тогда
        Команда = КомандыОтчетов.Добавить();
        …
    КонецЕсли;
КонецПроцедуры
```

Команды контекстных отчетов, выводимые сразу во всех (или во многих) объектах метаданных, рекомендуется описывать в процедуре `ПередДобавлениемКомандОтчетов` общего модуля `ВариантыОтчетовПереопределяемый`. Например:

```
Процедура ПередДобавлениемКомандОтчетов(КомандыОтчетов, Параметры, СтандартнаяОбработка) Экспорт
    Если ПравоДоступа("Просмотр", Метаданные.Отчеты.МестаИспользованияСсылок) Тогда
        Команда = КомандыОтчетов.Добавить();
        Команда.Представление      = НСтр("ru = 'Места использования'");
        Команда.МножественныйВыбор = Истина;
        Команда.ИмяПараметраФормы  = "Отбор.НаборСсылок";
        Команда.КлючВарианта       = "Основной";
        Команда.Менеджер           = "Отчет.МестаИспользованияСсылок";
    КонецЕсли;
КонецПроцедуры
```

Кроме того, команды отчетов можно задавать в самих отчетах конфигурации (и в отчетах расширений конфигурации). Для этого отчет нужно включить в состав подсистемы `ПодключаемыеОтчетыИОбработки` и в его модуле менеджера в области `ПрограммныйИнтерфейс` определить процедуры `ПриОпределенииНастроек` и `ДобавитьКомандыЗаполнения`. Подробнее см. раздел Подключение отчетов и обработок к механизмам конфигурации документации подсистемы Подключаемые команды. Пример:

```
#Область ПрограммныйИнтерфейс
// Определяет состав программного интерфейса для интеграции с конфигурацией.
//
// Параметры:
//   Настройки - Структура - Настройки интеграции этого объекта.
//       См. возвращаемое значение функции ПодключаемыеКоманды.НастройкиПодключаемыхОтчетовИОбработок().
//
Процедура ПриОпределенииНастроек(Настройки) Экспорт
    Настройки.Размещение.Добавить(Метаданные.Документы.ИмяДокумента);
    Настройки.ДобавитьКомандыОтчетов = Истина;
КонецПроцедуры
// Определяет список команд отчетов.
//
// Параметры:
//   КомандыОтчетов - ТаблицаЗначений - Таблица с командами отчетов. Для изменения.
//       См. описание 1 параметра процедуры ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов().
//   Параметры - Структура - Вспомогательные параметры. Для чтения.
//       См. описание 2 параметра процедуры ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов().
//
Процедура ДобавитьКомандыОтчетов(КомандыОтчетов, Параметры) Экспорт
КонецПроцедуры
#КонецОбласти
```

> **Примечание**
>
> Для команд отчетов, добавленных в процедуре `ДобавитьКомандыОтчетов`, подключенной указанным выше образом, заполнение параметра `Менеджер` не обязательно.

Поставка команд отчетов в отчетах конфигурации рекомендуется в тех случаях, когда команды обслуживают сразу несколько объектов (набор обслуживаемых объектов является группирующим признаком для таких команд).

Процедура `ДобавитьКомандыОтчетов` отчетов, подключенных по такой технологии, вызывается для всех объектов метаданных, указанных в параметре `Размещение` процедуры `ПриОпределенииНастроек`.

Колонки таблицы «КомандыОтчетов»

Параметр	Тип	Описание
`Идентификатор` (необязательный)	`Строка`.	Идентификатор команды, который используется для идентификации команды и определения имени команды в форме. Если идентификатор не указан, то имя команды будет определено автоматически.
`Представление` (обязательный)	`Строка`.	Представление команды в форме. Пример: `Команда.Представление = НСтр("ru = 'Заполнить ТТН'");`
`Важность` (необязательный)	`Строка`.	Краткое имя (суффикс) группы в подменю, в которой следует вывести эту команду. Допустимо использовать: `Важное`, `Обычное` и `СмТакже`.
`Порядок` (необязательный)	`Число`.	Значение от 1 до 100, указывающее порядок размещения команды по отношению к другим командам. Сортировка команд осуществляется сначала по полю `Порядок`, затем по представлению. Значение по умолчанию: 50
`СочетаниеКлавиш` (необязательный)	`СочетаниеКлавиш`.	Сочетание клавиш для быстрого вызова команды.

Колонки таблицы «КомандыОтчетов», настройки видимости

Параметр	Тип	Описание
`ТипПараметра` (необязательный)	`ОписаниеТипов`.	Типы объектов, для которых предназначена эта команда. Используется, когда обработка (или другой поставщик команд) подключена к нескольким объектам конфигурации, для уточнения списка объектов, к которым подключена конкретная команда.
`ВидимостьВФормах` (необязательный)	`Строка`.	Имена форм (через запятую), в которых должна отображаться команда. Используется для уточнения состава форм, к которым подключена конкретная команда. Если параметр не указан, то команда будет отображаться во всех формах. Пример: `Команда.ВидимостьВФормах = "ФормаДокумента"`
`ФункциональныеОпции` (необязательный)	`Строка`.	Имена функциональных опций (через запятую), определяющих видимость команды.
`УсловияВидимости` (необязательный)	`Массив`.	Определяет видимость команды в зависимости от контекста. Для регистрации условий следует использовать процедуру `ПодключаемыеКоманды.ДобавитьУсловиеВидимостиКоманды`. Условия объединяются по «И».
`ИзменяетВыбранныеОбъекты` (необязательный)	`Булево`.	Определяет доступность команды в ситуации, когда у пользователя нет прав на изменение объекта. Если `Истина`, то кнопка будет недоступна. Значение по умолчанию: `Ложь`.
`Неконтекстный` (необязательный)	`Булево`.	Если `Истина`, то будут доступны для выбора другие варианты этого отчета, возможность добавления в избранное, получение ссылки на вариант отчета. Значение по умолчанию: `Ложь`.

Колонки таблицы «КомандыОтчетов», настройки процесса выполнения

Параметр	Тип	Описание
`МножественныйВыбор` (необязательный)	`Булево`.	Если `Истина`, то команда поддерживает множественный выбор и в 1 параметре обработчика команды будет передан массив ссылок. Если `Ложь`, то команда поддерживает только одиночный выбор и в 1 параметре обработчика команды будет передана 1 ссылка.
`РежимЗаписи` (необязательный)	`Строка`.	Настройки дополнительных проверок и действий, связанных с записью объекта (выполняются перед обработчиком команды). • `НеЗаписывать` – объект не записывается, а в параметрах обработчика вместо ссылок передается вся форма. В этом режиме рекомендуется работать напрямую с формой, которая передается в структуре 2-го параметра обработчика команды. • `ЗаписыватьТолькоНовые` – записывать только новые объекты. • `Записывать` – записывать новые и модифицированные объекты. • `Проводить` – проводить документы. Пример: `Команда.РежимЗаписи = "НеЗаписывать";` Перед записью и проведением у пользователя запрашивается подтверждение. Значение по умолчанию: `Записывать`.
`ТребуетсяРаботаСФайлами` (необязательный)	`Булево`.	Если `Истина`, то в веб-клиенте перед выполнением команды предлагается установить расширение для работы с 1С:Предприятием.

Колонки таблицы «КомандыОтчетов», настройки обработчика

Параметр	Тип	Описание
`Менеджер` (необязательный)	`Строка`.	Полное имя отчета, отвечающего за выполнение команды. Если `ИмяФормы` и `Обработчик` не указаны, то открывается основная форма отчета с параметрами, соответ режиму работы отчета (их дублирование в структуре `ПараметрыФормы` не требуется).
`ИмяФормы` (обязательный если не указан `Обработчик`)	`Строка`.	Имя формы, которую требуется открыть или получить для выполнения команды. Если `Обработчик` не указан, то у формы вызывается метод `Открыть`.
`КлючВарианта` (необязательный)	`Строка`.	Имя варианта отчета, открываемого при выполнении команды.
`ИмяПараметраФормы` (необязательный)	`Строка`.	Имя параметра формы, в который следует передать ссылку или массив ссылок.
`ПараметрыФормы` (необязательный)	`Структура`, `Неопределено`.	Параметры формы, указанной в `ИмяФормы`.

`Обработчик` (обязательный если не указано `ИмяФормы`)	`Строка`.	Имя процедуры, обрабатывающей основное действие команды.
		Пример № 1 – обработчик размещен в общем модуле:
		`Команда.Обработчик = "ПродажиКлиент.ВвестиСчетФактуру";` Пример № 2 – обработчик размещен в модуле ф
		`Команда.Обработчик = "ЗаполнитьТТН";` Если `ИмяФормы` заполнено, то в модуле указанной формы ожидаетс по шаблону:
		`&НаКлиенте Процедура <ИмяПроцедуры>(<ИмяПараметраКоманды>, ПараметрыВыполнения) Экспорт // Код обработчи`
		Если `ИмяФормы` не заполнено, то в модуле менеджера объекта, указанного в `Менеджер`, ожидается сервер шаблону:
		`Процедура <ИмяПроцедуры>(<ИмяПараметраКоманды>, ПараметрыВыполнения) Экспорт // Код обработчика команды.`
		параметра обработчика команды `ИмяПараметраКоманды` рекомендуется задавать в соответствии с типами предназначена команда. Также следует учитывать, что тип этого параметра зависит от настройки `Множест` `МножественныйВыбор` = `Истина`, тогда передается массив ссылок, в противном случае передается ссылка.
		предназначена для заполнения документов «Счет на оплату» и `МножественныйВыбор` включен, то параметр `МассивСчетовНаОплату` или `СчетаНаОплату`. Если `МножественныйВыбор` отключен, то параметр можно назвать `ССылкаСчетаНаОплату`.
		2-й параметр обработчика команды `ПараметрыВыполнения` имеет тип `Структура` со следующими полями:
		• `ОписаниеКоманды` – `Структура` – описание команды (ключи соответствуют колонкам этой таблицы).
		• `Идентификатор` – `Строка` – идентификатор команды.
		• `Представление` – `Строка` – представление команды в форме.
		• `ДополнительныеПараметры` – `Неопределено`, `ФиксированнаяСтруктура` – дополнительные параметры ком
		• `Форма` – `ФормаКлиентскогоПриложения` – форма, из которой вызвана команда.
		• `ЭтоФормаОбъекта` – `Булево` – `Истина`, если команда вызвана из формы объекта.
		• `Источник` – `ТаблицаФормы`, `ДанныеФормыСтруктура` – объект или список формы с полем `Ссылка`.

`ДополнительныеПараметры` (необязательный)	`Структура`, `Неопределено`.	Дополнительные параметры, которые могут быть считаны в обработчике команды.

См. также пример реализации команды контекстного отчета в отчете расширения **Демо: Подключаемые команды** в демонстрационной базе.

Если контекстные отчеты не требуется выводить в панелях отчетов и в списках отчетов, то при настройке вариантов этих отчетов следует установить им в свойстве `Включен` значение `Ложь`.

Включение контекстной настройки отчетов

В отчетах на базе системы компоновки данных (СКД) при нажатии на колонку доступны возможности контекстного меню и настройки свойств колонки. При этом если в конфигурации для отчетов определен свой макет оформления компоновки данных, то для включения этих возможностей необходимо внести в него изменения: для заголовков таблиц и группировок установить режим изменения размера колонки **Быстрое изменение**. См. пример в демонстрационной конфигурации в общем макете `_ДемоОформлениеОтчетовБежевый` и в процедуре `ПередЗагрузкойВариантаНаСервере` общего модуля `ОтчетыПереопределяемый`.

Особенности работы в мобильном клиенте в автономном режиме

Для просмотра снимков вариантов отчетов в мобильном клиенте в автономном режиме (при отсутствии соединения с сервером) необходимо настроить выгрузку данных в автономную конфигурацию:

1. Создать и включить в состав автономной конфигурации план обмена, в составе которого указать справочники `ВариантыОтчетов`, `Пользователи` и регистр сведений `СнимкиОтчетов`. См. пример в демонстрационной конфигурации в плане обмена `_ДемоМобильныйКлиент`.
2. Настроить регистрацию в плане обмена при записи указанных выше объектов, например, через подписку на событие `ПриЗаписи`. См. пример в демонстрационной конфигурации в подписках на события `_ДемоЗарегистрироватьИзмененияДляАвтономногоРежима` и `_ДемоЗарегистрироватьИзмененияДляАвтономногоРежимаРегистры`.
3. Настроить запуск обмена данными в фоновом режиме с включением обработчика ожидания при начале сеанса мобильного клиента при доступности основного сервера - см. пример в демонстрационной конфигурации общий модуль `ОбщегоНазначенияКлиентПереопределяемый` процедура `ПриНачалеРаботыСистемы`.

4. Включить в состав автономной конфигурации все объекты метаданных (например, общие модули), использующиеся в алгоритме выгрузки и загрузки данных в автономную конфигурацию - см. пример в демонстрационной конфигурации общий модуль `_ДемоОбменМобильныйКлиентАвтономныйСервер`
5. Добавить роль, предоставляющую полные права на работу с планом обмена для обмена с автономной конфигурацией, см. пример в демонстрационной конфигурации роль `_ДемоОбменМобильныйКлиент`. Также в эту роль включить права для работы с объектами метаданных, участвующими в обмене, см. пример в автономной конфигурации константу `_ДемоКодНовогоУзлаПланаОбмена`. Добавить данную роль, а также роль `ДобавлениеИзменениеСнимковОтчетов` в профили групп доступа пользователей, работающих в мобильном клиенте.

Особенности при работе в модели сервиса

Если конфигурация рассчитана на работу в модели сервиса, то для некоторых отчетов может потребоваться определить функцию `ВариантыНастроек` в модуле менеджера отчета. Такая необходимость возникает в тех случаях, когда схема компоновки данных содержит ссылки на предопределенные элементы справочников или других таблиц, разделенных основным разделителем `ОбластьДанныхОсновныеДанные`. Поскольку в модели сервиса сбор сведений о предопределенных вариантах отчетов конфигурации выполняется в сеансе без установленных значений разделителей, то после неудачной попытки автоматического анализа схемы компоновки данных вызывается данная функция.

Пример реализации см. в модуле менеджера отчета `_ДемоДинамикаИзмененийФайлов` демонстрационной конфигурации:

```
#Область ПрограммныйИнтерфейс
#Область ДляВызоваИзДругихПодсистем
// СтандартныеПодсистемы.ВариантыОтчетов
// Вызывается при работе в модели сервиса для получения сведений о предопределенных вариантах отчета.
//
// Возвращаемое значение:
//  Массив - сведения о вариантах отчета, Структура со свойствами:
//    * Имя           - Строка - имя варианта отчета; например, "Основной";
//    * Представление - Строка - имя варианта отчета; например, НСтр("ru = 'Динамика изменений файлов'").
//
Функция ВариантыНастроек() Экспорт
    Результат = Новый Массив;
    Результат.Добавить(Новый Структура("Имя, Представление", "Основной", НСтр("ru = 'Динамика изменений файлов'")));
    Результат.Добавить(Новый Структура("Имя, Представление", "Дополнительный1", НСтр("ru = ' Динамика изменений файлов по авторам'")));
    Возврат Результат;
КонецФункции
// Конец СтандартныеПодсистемы.ВариантыОтчетов
#КонецОбласти
#КонецОбласти
```

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме:

- справочника `ПредопределенныеВариантыОтчетовРасширений`,
- регистра сведений `ПредопределенныеВариантыОтчетовВерсийРасширений`,
- регистра сведений `СнимкиОтчетов`.

Также их рекомендуется исключать из остальных планов обмена, предназначенных для синхронизации данных между приложениями.

Версионирование объектов

Подсистема **Версионирование объектов** предназначена для учета истории изменений объектов (кто, когда и что изменил). Также она позволяет получать отчеты по версиям или по конкретной версии объекта. Версионируемыми объектами могут быть справочники, документы, бизнес-процессы, планы видов характеристик, планы счетов конфигурации и планы видов расчета.

Настройка

Для использования подсистемы в конфигурации необходимо:

- если в конфигурации не используется подсистема **Настройки программы**, то разместить в командном интерфейсе администратора приложения регистр сведений `НастройкиВерсионированияОбъектов` и поместить константу `ИспользоватьВерсионированиеОбъектов` в основную форму редактирования констант конфигурации или в любую другую форму, предназначенную для администрирования системы;
- создать подписку `ЗаписатьВерсиюДокумента` на событие `ПередЗаписью`, обработчик `ВерсионированиеОбъектовСобытия.ЗаписатьВерсиюДокумента`.

Для размещения настроек подсистемы в произвольной форме следует:

- создать реквизит формы `ХранитьИсториюИзменений` типа `Булево` для управления флажком настроек, который следует разместить в элементах формы;
- в обработчике события формы `ПриСозданииНаСервере` вызвать процедуру `ЗначениеФлажкаХранитьИсторию` модуля `ВерсионированиеОбъектов`.
- в обработчике события формы `ОбработкаОповещения` вызвать процедуру `ОбработкаОповещенияИзмененияФлажкаХранитьИсторию` модуля `ВерсионированиеОбъектовКлиент`.
- при изменении реквизита формы `ХранитьИсториюИзменений` вызвать процедуру `ПриИзмененииФлажкаХранитьИсторию` модуля `ВерсионированиеОбъектовКлиент`.

Принять решение по поводу объектов метаданных конфигурации ссылочного типа, которые следует версионировать, затем:

- все версионируемые объекты перечислить в свойстве `Тип` определяемого типа `ВерсионируемыеДанные` (типы `Ссылка` – например, `СправочникСсылка` или `ДокументСсылка`);
- в свойстве `Тип` определяемого типа `ВерсионируемыеДанныеОбъект` перечислить все версионируемые объекты, кроме документов (типы `Объект` – например, `СправочникОбъект` или `БизнесПроцессОбъект`);
- перечислить все версионируемые документы в свойстве `Источник` подписок `ЗаписатьВерсиюДокумента` (типы `ДокументОбъект`);
- во всех формах версионируемых объектов и документов в обработчике `ПриСозданииНаСервере` добавить фрагмент кода:

```
// СтандартныеПодсистемы.ВерсионированиеОбъектов
ВерсионированиеОбъектов.ПриСозданииНаСервере(ЭтотОбъект);
// Конец СтандартныеПодсистемы.ВерсионированиеОбъектов
```

- во всех модулях менеджера объектов (элементов), для которых встраивается версионирование, добавить фрагмент кода:

```
// СтандартныеПодсистемы.ВерсионированиеОбъектов
// Определяет настройки объекта для подсистемы ВерсионированиеОбъектов.
//
// Параметры:
// Настройки - Структура - настройки подсистемы.
Процедура ПриОпределенииНастроекВерсионированияОбъектов(Настройки) Экспорт
КонецПроцедуры
// Конец СтандартныеПодсистемы.ВерсионированиеОбъектов
```

- принять решение о необходимости скрытия из отчетов по версиям реквизитов и табличных частей, имеющих техническое назначение. Для скрытия реквизитов и табличных частей необходимо в процедуре `ПриОпределенииНастроекВерсионированияОбъектов` модуля менеджера объекта добавить следующий код:

```
    Настройки.ПриПолученииСлужебныхРеквизитов = Истина;
```

  и добавить в модуле менеджера процедуру `ПриПолученииСлужебныхРеквизитов`, в которой перечислить список скрываемых реквизитов и табличных частей:

```
// СтандартныеПодсистемы.ВерсионированиеОбъектов
// Определяет настройки объекта для подсистемы ВерсионированиеОбъектов.
//
// Параметры:
//  Настройки - Структура - настройки подсистемы.
Процедура ПриОпределенииНастроекВерсионированияОбъектов(Настройки) Экспорт
    Настройки.ПриПолученииСлужебныхРеквизитов = Истина;
КонецПроцедуры
// Ограничивает видимость реквизитов объекта в отчете по версии.
//
// Параметры:
//  Реквизиты - Массив - список имен реквизитов объекта.
Процедура ПриПолученииСлужебныхРеквизитов(Реквизиты) Экспорт
    Реквизиты.Добавить("ИмяРеквизита"); // реквизит объекта
    Реквизиты.Добавить("ИмяТабличнойЧасти.*"); // табличная часть объекта
КонецПроцедуры
// Конец СтандартныеПодсистемы.ВерсионированиеОбъектов
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Версионирование объектов** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение использования версионирования. Чтение версий объектов. Просмотр отчета по версиям. Возврат к предыдущим версиям объектов. Изменение настроек хранения объектов. Очистка устаревших версий объектов.
2.	`ЧтениеВерсийОбъектов` Просмотр версий объектов и отчета по версиям.

Использование при разработке конфигурации

Разработка макетов отчета по конкретной версии объекта

Подсистема **Версионирование объектов** позволяет использовать произвольный макет отчета по версии объекта вместо макета по умолчанию.

Для создания макета отчета для версионируемого объекта можно воспользоваться конструктором печати объекта:

- открыть конструктор печати объекта;
- выбрать **Создать новую команду**, ввести имя команды (например, **Печать**; при этом необходимо убедиться, что процедуры с таким именем нет в модуле менеджера объекта метаданных) и нажать кнопку **Далее**;
- выбрать необходимые реквизиты, которые будут выводиться в отчете, и нажать кнопку **Далее**;
- выбрать необходимые поля в табличных частях объекта (если есть) и последовательно нажать кнопку **Далее** для каждой табличной части;
- не выбирать реквизиты для подвала документа и нажать кнопку **Далее**;

- в итоговом диалоге выбора группы оставить все значения по умолчанию и нажать **ОК**;
- удалить созданную команду печати и процедуру формирования данных печати из модуля менеджера объекта, которые были созданы на предыдущем шаге (в примере **Печать**);
- переименовать созданный макет в `МакетОбъекта` и при необходимости настроить его оформление.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Взаимодействия

Подсистема **Взаимодействия** предназначена для планирования, регистрации, упорядочивания взаимодействий и работы с результатами взаимодействий. Взаимодействия включают переписку по электронной почте, регистрацию звонков и встреч. Подсистема обеспечивает подбор и создание новых контактов взаимодействий.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то на рабочем месте администратора приложения необходимо разместить константы:

- `ИспользоватьПочтовыйКлиент`,
- `ОтправлятьПисьмаВФорматеHTML`,
- `ИспользоватьПрочиеВзаимодействия`,
- `ИспользоватьПризнакРассмотрено`.

См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

В форме персональных настроек разместить вызов команды `ЖурналДокументов.Взаимодействия.Команда.НастройкиРаботыСПочтой`. Пример размещения см. в демонстрационной конфигурации в общей форме `_ДемоМоиНастройки`.

Разместить в командном интерфейсе пользователей, использующих подсистему, журнал документов **Взаимодействия**.

Определение типов предметов и контактов взаимодействий

Необходимо принять решение по поводу состава объектов метаданных, которые могут выступать в качестве контактов взаимодействий. Например, это могут быть физические лица, партнеры, контактные лица партнеров и т. п. Расширить тип определяемого типа `КонтактВзаимодействия` допустимыми типами контактов.

Если предусматривается создание контактов напрямую из форм документов взаимодействий, то для форм объектов метаданных, определенных как контакты, необходимо:

- добавить реквизит формы `НеобходимоОповещение` типа `Булево`,
- добавить реквизит формы `ОбъектОснование` типа `Произвольный`,
- добавить в обработчик события `ПриСозданииНаСервере` вызов следующей процедуры:

```
Взаимодействия.ПодготовитьОповещения(ЭтотОбъект, Параметры, Ложь);
```

- добавить в обработчик события формы `ПослеЗаписи` вызов следующей процедуры:

```
ВзаимодействияКлиент.КонтактПослеЗаписи(Форма, Объект, ПараметрыЗаписи, ИмяОбъектаОтправителяСообщения);
```

Необходимо принять решение по поводу состава объектов метаданных, которые могут выступать в качестве предметов взаимодействий. Например, это могут быть проекты, заказы клиентов, заказ поставщику и т. п. Расширить тип определяемого типа `ПредметВзаимодействия` допустимыми типами предметов.

Если предусматривается создание предметов на основании документов взаимодействий, то для форм объектов метаданных, определенных как контакты, необходимо:

- добавить реквизит формы `НеобходимоОповещение` типа `Булево`;
- добавить реквизит формы `ВзаимодействиеОснование` составного типа, включающего в себя типы тех документов взаимодействий, на основании которых создаются предметы взаимодействий;
- добавить в обработчик события `ПриСозданииНаСервере` вызов следующей процедуры:

```
Взаимодействия.ПодготовитьОповещения(ЭтотОбъект,Параметры,Ложь);
```

- добавить в обработчик события формы `ПослеЗаписи` вызов следующей процедуры:

```
ВзаимодействияКлиент.ВзаимодействиеПредметПослеЗаписи(Форма, Объект, ПараметрыЗаписи, ИмяОбъектаОтправителяСообщения);
```

Настройка модуля менеджера предметов взаимодействий

В модуле менеджера каждого объекта метаданных, определенного как «предмет взаимодействий», необходимо реализовать экспортную функцию `ТекстЗапросаПоКонтактам`, в которой сформировать текст запроса по контактам, содержащимся в предмете взаимодействий. Например, ссылки на контакты взаимодействий могут иметься в реквизитах шапки и табличных частей предметов взаимодействий.

Функция принимает два необязательных параметра. В первом параметре `ТекстВременнаяТаблица` типа `Строка` может находиться часть текста запроса, отвечающая за помещение результата запроса во временную таблицу. Второй параметр `Объединить` типа `Булево` указывает на режим формирования запроса. Если данный параметр имеет значение `Истина`, то формируемый в функции запрос является частью другого запроса и должен начинаться с конструкции `ОБЪЕДИНИТЬ`. Возвращаемое значение – `Строка`, содержащая в себе текст запроса по контактам предмета взаимодействий.

Также в модуле менеджера каждого объекта метаданных, определенного как «предмет взаимодействий», необходимо реализовать экспортную функцию `ТекстЗапросаПоКонтактам`, в которой будет сформирован массив возможных контактов предмета. Функция принимает обязательный параметр – ссылку на объект, содержащий контакты. Возвращаемое значение – `Массив`, содержащий в себе возможные контакты документа взаимодействий по предмету.

Пример реализации данных функций можно посмотреть в модуле менеджера документа `_ДемоЗаказПокупателя` в демонстрационной конфигурации.

Для тех предметов взаимодействий, для которых предполагается использовать механизм активности предметов и текущий статус подразумевает активность предмета, необходимо устанавливать признак активности в регистр сведений `СостоянияПредметовВзаимодействий`. Например, как это сделано в процедуре `ПриЗаписи` модуля объекта документа `_ДемоЗаказПокупателя`.

Настройка переопределяемых модулей

При необходимости можно вписать реализацию в функции переопределяемых модулей подсистемы:

- `ВзаимодействияКлиентПереопределяемый`,
- `ВзаимодействияКлиентСерверПереопределяемый`,
- `ВзаимодействияПереопределяемый`.

Настройка форм списков предметов взаимодействий.

Для того чтобы появилась возможность устанавливать предмет взаимодействий путем множественного перетаскивания в форму списка предмета, необходимо выполнить следующие действия:

- у элемента формы списка установить флажок `РазрешитьПеретаскивание`;
- в обработчике события `ПроверкаПеретаскивания` элемента формы списка разместить следующий код:

```
ВзаимодействияКлиент.СписокПредметПроверкаПеретаскивания(Элемент, ПараметрыПеретаскивания, СтандартнаяОбработка, Строка, Поле);
```

- в обработчике события `Перетаскивание` элемента формы списка разместить следующий код:

```
ВзаимодействияКлиент.СписокПредметПеретаскивание(Элемент, ПараметрыПеретаскивания, СтандартнаяОбработка, Строка, Поле);
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Взаимодействия** следует использовать роль, приведенную ниже.

№	Роли и их назначение
1.	`ДобавлениеИзменениеВзаимодействий`. Чтение, добавление, изменение взаимодействий.

Пример настройки прав доступа пользователей приведен ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за ведение взаимодействий.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты` (из подсистемы **Работа с почтовыми сообщениями**), • `ДобавлениеИзменениеВзаимодействий`.

Особые случаи внедрения подсистемы

- Внедрение подсистем «Взаимодействия» и «Управление доступом».
- Внедрение подсистемы «Взаимодействия» без подсистемы «Электронная подпись».

Использование при разработке конфигурации

Настройка обмена данными

Все объекты метаданных подсистемы рекомендуется включать в планы обмена распределенной информационной базы (РИБ), автономных рабочих мест, а также в планы обмена для синхронизации данных между различными приложениями, за исключением следующих регистров сведений:

- `ЗаблокированныеДляПолученияУчетныеЗаписи` - предназначен для предотвращения дублирования получаемых писем при конкурентной работе с общими учетными записями,
- `СостоянияПапокПисем`,
- `СостоянияПредметовВзаимодействий`,
- `СостоянияКонтактовВзаимодействий`.

В последних трех регистрах хранятся сводные сведения о документах взаимодействий - количество нерассмотренных и дата последнего взаимодействия, которые выводятся в рабочем месте ответственного за ведение взаимодействия. Для того чтобы эти данные были корректными, при завершении обмена необходимо выполнить перерасчет при помощи следующих процедур:

- `Взаимодействия.РассчитатьРассмотреноПоКонтактам`,
- `Взаимодействия.РассчитатьРассмотреноПоПапкам`,
- `Взаимодействия.РассчитатьРассмотреноПоПредметам`.

Внешние компоненты

Подсистема **Внешние компоненты** предназначена для загрузки в приложение внешних компонент сторонних разработчиков. Внешние компоненты расширяют возможности системы **1С:Предприятие 8**, позволяют подключать торговое оборудование, выполнять взаимодействие с операционной системой и сторонними приложениями, обеспечивают возможность взаимодействия с внешними источниками данных.

При этом администратор приложения контролирует состав и версии внешних компонент, с которыми разрешена работа пользователям приложения. Подсистема централизованно обеспечивает загрузку, обновление, установку и подключение внешних компонент. Кроме того, при установке и подключении внешних компонент выполняется контроль возможности использовать внешнюю компоненту в текущем приложении (например, если компонента предназначена для тонкого клиента, но не работает в веб-клиенте). Загрузка внешних компонент может выполняться как автоматически с портала 1С:ИТС (при наличии подсистемы **Получение внешних компонент** библиотеки **Библиотека интернет-поддержки**), так и администратором вручную из указанного файла.

Для работы подсистемы в модели сервиса необходимо дополнительно встроить в конфигурацию подсистему **Внешние компоненты в модели сервиса**.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить команду открытия списка справочника `ВнешниеКомпоненты`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Внешние компоненты** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**) Получение внешних компонент для установки.
2.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Администрирование внешних компонент.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор.	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Пользователь.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**).
3.	Внешний пользователь.	• `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускВебКлиента` (из подсистемы **Базовая функциональность**).

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Доступ к внешним компонентам непосредственно из справочника `Справочник.ВнешниеКомпоненты` не предусмотрен.

Пример подключения внешней компоненты для использования сканера штрихкодов:

```
//     Подключение внешней компоненты
    Оповещение = Новый ОписаниеОповещения("ПодключитьКомпонентуЗавершение", ЭтотОбъект);

    ПараметрыПодключения = ВнешниеКомпонентыКлиент.ПараметрыПодключения();
    ПараметрыПодключения.ТекстПояснения =
        НСтр("ru = 'Демо: Для использования сканера штрихкодов требуется
                    |внешняя компонента «1С:Сканеры штрихкода (NativeApi)».'");

    ВнешниеКомпонентыКлиент.ПодключитьКомпоненту(Оповещение, "InputDevice",, ПараметрыПодключения);
// Конец Подключение внешней компоненты
// При оповещении о подключении
&НаКлиенте
Процедура ПодключитьКомпонентуПослеПодключения(Результат, ДополнительныеПараметры) Экспорт

    ПодключаемыйМодуль = Неопределено;

    Если Результат.Подключено Тогда
        ПодключаемыйМодуль = Результат.ПодключаемыйМодуль;
    Иначе
        Если Не ПустаяСтрока(Результат.ОписаниеОшибки) Тогда
            ПоказатьПредупреждение(, Результат.ОписаниеОшибки);
        КонецЕсли;
    КонецЕсли;

    Если ПодключаемыйМодуль <> Неопределено Тогда
        ПоказатьОповещениеПользователя(
            НСтр("ru = 'Демо: Успешное подключение'"),,,
            БиблиотекаКартинок.Успешно32);
    КонецЕсли;

    ПодключаемыйМодуль = Неопределено;

КонецПроцедуры
// Конец При оповещении о подключении
```

В случае, когда требуется привязать определенную внешнюю компоненту к системным настройкам (например, справочник `НастройкиОбменСБанками` в составе БЭД или справочник `ДрайверыОборудования` в составе БПО), следует предусмотреть сохранение значения `Идентификатор`, определяющего компоненту (и `Версия`, если требуется компонента конкретной версии). Выполнять получение информации о компоненте следует с помощью функции `ИнформацияОКомпоненте` общего модуля `ВнешниеКомпонентыВызовСервера`. При необходимости можно реализовать загрузку и обновление компоненты в форме системных настроек из файла на компьютере. Пример представлен в обработке `_ДемоРаботаСВнешнимиКомпонентами` в демонстрационной конфигурации.

Настройка обмена данными

Все объекты метаданных подсистемы требуется исключить из планов обмена, предназначенных для синхронизации данных между приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места.

Генерация штрихкода

Подсистема предоставляет программный интерфейс для генерирования изображений штрихкодов EAN8, EAN13, EAN128, Code39, Code93, Code128, Code16k, PDF417, ITF14, RSS14, EAN13AddOn2, EAN13AddOn5, QR, GS1DataBarExpandedStacked и Datamatrix.

Для генерации изображения штрихкода используется внешняя компонента, хранящаяся в общем макете КомпонентаПечатиШтрихкодов, которая будет загружена и инициализирована автоматически. В случае если внедрена подсистема **Внешние компоненты**, администратор может обновлять версию компоненты с помощью справочника **Внешние компоненты**.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей дополнительных ролей не требуется.

Использование при разработке конфигурации

Программный интерфейс подсистемы представлен в общем модуле `ГенерацияШтрихкода`. Пример генерации изображения штрихкода:

```
ПараметрыШтрихкода = ГенерацияШтрихкода.ПараметрыГенерацииШтрихкода();
ПараметрыШтрихкода.Ширина = ШиринаШтрихкода;
ПараметрыШтрихкода.Высота = ВысотаШтрихкода;
ПараметрыШтрихкода.ТипКода = 1; // EAN13
ПараметрыШтрихкода.УголПоворота = 0;
ПараметрыШтрихкода.Штрихкод = ЗначШтрихкод;
ПараметрыШтрихкода.ПрозрачныйФон = Истина;
ПараметрыШтрихкода.Масштабировать = Истина;
ПараметрыШтрихкода.СохранятьПропорции = Истина;
ПараметрыШтрихкода.ВертикальноеВыравнивание = ВертикальноеВыравнивание;
ПараметрыШтрихкода.GS1DatabarКоличествоСтрок = КоличествоСтрокGS1Databar;
ПараметрыШтрихкода.ТипВходныхДанных = 0;  // Тип входных данных (0-Строка, 1-Base64)

РезультатШтрихкод = ГенерацияШтрихкода.ИзображениеШтрихкода(ПараметрыШтрихкода);
```

См. также в демонстрационной конфигурации пример вывода изображения QR-кода в печатную форму в модуле менеджера `_ДемоСчетНаОплатуПокупателю`.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Графики работы

Подсистема **Графики работы** предназначена для хранения графиков, по которым работает предприятие. Для каждого графика можно указать, какие дни являются рабочими. Графики работы можно использовать как для указания работы компании в целом, так и отдельных ее частей. Например, график работы склада по выдаче товаров.

Настройка

В командном интерфейсе ответственного за ведение нормативно-справочной информации необходимо поместить справочник **Календари**. См. пример в демонстрационной конфигурации в разделе `Справочники`.

Возможно определить принадлежность графиков работы к отдельным объектам, участвующим в бизнес-логике. Например, можно связать график работы с конкретным подразделением, складом или даже станком. Для использования этой возможности, достаточно включить типы объектов, имеющих собственные графики работы, в определяемый тип `ВладелецГрафиковРаботы`. Графики работы, для которых указан владелец, будут скрыты из общего списка графиков.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Графики работы** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ЧтениеГрафиковРаботы` (из подсистемы **Базовая функциональность**) Просмотр календарей и графиков.
2.	`ДобавлениеИзменениеГрафиковРаботы` Добавление и изменение календарей и графиков.
3.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Удаление помеченных на удаление объектов подсистемы.

Использование при разработке конфигурации

Подсистема используется совместно с подсистемой **Календарные графики**.

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Групповое изменение объектов

Подсистема **Групповое изменение объектов** предоставляет пользовательский интерфейс для множественного изменения объектов одного типа. Помимо изменения реквизитов имеется возможность изменения дополнительных реквизитов и сведений (подсистема **Свойства**), если они есть у объектов.

Также подсистема учитывает наличие у объектов реквизитов, запрещенных для редактирования (подсистема **Запрет редактирования реквизитов объектов**).

Настройка

Размещение команды изменения объектов в списках

Сначала нужно принять решение по поводу состава объектов конфигурации, для которых должна быть добавлена команда по групповому изменению. Как правило, к таким объектам относятся большинство объектов конфигурации ссылочного типа.

При использовании подсистемы Подключаемые команды существует возможность добавить команду в списки и журналы. Для этого необходимо перечислить объекты в процедуре `ПриОпределенииОбъектовСКомандойГрупповогоИзмененияОбъектов` общего модуля `ГрупповоеИзменениеОбъектовПереопределяемый`.

Также можно разместить самостоятельно команду для группового изменения объектов на командной панели и в контекстном меню формы списка. Рекомендуемые значения свойств команды:

Свойство	Значение
Имя	`ИзменитьВыделенные`.
Действие	`ИзменитьВыделенные`.
Подсказка	Изменить выделенные объекты.
Изменяет сохраняемые данные	Нет.

Также рекомендуется управлять видимостью команды в форме в зависимости от текущих прав пользователя. Например:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    МожноРедактировать = ПравоДоступа("Редактирование", Метаданные.Справочники._ДемоНоменклатура);
    Элементы.ФормаИзменитьВыделенные.Видимость = МожноРедактировать;
КонецПроцедуры
```

Установить у поля `Ссылка` основного реквизита формы признак **Использовать всегда**.

Добавить в модуль формы обработчик:

```
&НаКлиенте
Процедура ИзменитьВыделенные(Команда)

    ГрупповоеИзменениеОбъектовКлиент.ИзменитьВыделенные(Элементы.Список, Список);

КонецПроцедуры
```

Ограничение редактируемых реквизитов объектов

Изменение некоторых реквизитов с помощью групповой обработки может быть нежелательным либо нарушать логику работы конфигурации. Такие реквизиты необходимо скрывать от пользователя.

Обработка группового изменения может быть вызвана как из формы списка для выбранных объектов, так и самостоятельно с последующим выбором изменяемых объектов в самой обработке, поэтому необходимо для всех объектов конфигурации рассмотреть необходимость ограничения доступности реквизитов для группового изменения.

Если для объекта метаданных определены реквизиты, которые можно изменять с помощью обработки группового изменения, следует объявить экспортную функцию `РеквизитыРедактируемыеВГрупповойОбработке` в модуле менеджера объекта, которая возвращает массив – имена реквизитов. При этом оставшиеся реквизиты скрываются в обработке группового изменения. В общем виде разрешенные для редактирования реквизиты объекта описываются следующим образом (пример см. в бизнес-процессе `_ДемоЗаданиеСРолевойАдресацией` демонстрационной конфигурации):

```
// Возвращает реквизиты объекта, которые разрешается редактировать
// с помощью обработки группового изменения реквизитов.
//
// Возвращаемое значение:
//  Массив - список имен реквизитов объекта.
Функция РеквизитыРедактируемыеВГрупповойОбработке() Экспорт
    РедактируемыеРеквизиты = Новый Массив;
    РедактируемыеРеквизиты.Добавить("ИмяРеквизита"); // реквизит объекта
    РедактируемыеРеквизиты.Добавить("ИмяТабличнойЧасти.ИмяРеквизита"); // реквизит табличной части объекта
    РедактируемыеРеквизиты.Добавить("ИмяТабличнойЧасти.*"); // все реквизиты табличной части объекта
    РедактируемыеРеквизиты.Добавить("*"); // все реквизиты и табличные части объекта
    Возврат РедактируемыеРеквизиты;
КонецФункции
```

Если стоит обратная задача: известны имена реквизитов, которые не должны редактироваться в обработке группового изменения, – следует объявить в модуле менеджера объекта функцию `РеквизитыНеРедактируемыеВГрупповойОбработке`, которая возвращает массив – имена реквизитов (аналогично функции `РеквизитыРедактируемыеВГрупповойОбработке`). Пример см. в справочнике `_ДемоНоменклатура` демонстрационной конфигурации.

Если объект не предназначен для использования в групповом изменении, то необходимо скрывать все его реквизиты.

> **Важно!**
>
> Служебные реквизиты объектов, такие как `Ссылка`, `Наименование`, `ПометкаУдаления` и т. д., а также некоторые табличные части не требуют дополнительных действий для скрытия в обработке группового изменения, так как они скрыты по умолчанию. Подробнее со списком таких реквизитов и табличных частей можно ознакомиться в макете `ФильтрРеквизитов` обработки `ГрупповоеИзменениеОбъектов`.

При необходимости также следует перенести логику проверки заполнения объектов и логику «при записи» из модулей форм в модули объектов.

Все объекты метаданных, в которых определены функции `РеквизитыРедактируемыеВГрупповойОбработке` и `РеквизитыНеРедактируемыеВГрупповойОбработке`, следует перечислить в процедуре `ПриОпределенииОбъектовСРедактируемымиРеквизитами` общего модуля `ГрупповоеИзменениеОбъектовПереопределяемый`.

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` Редактирование любых объектов конфигурации (при открытии формы группового изменения реквизитов из раздела командного интерфейса **Администрирование**).
2.	**Другие роли** Для группового изменения реквизитов в списках объектов не требуются какие-либо дополнительные роли. Эта возможность будет доступна, если у пользователя есть права на редактирование самих объектов.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Даты запрета изменения

Подсистема **Даты запрета изменения** позволяет заблокировать изменения любых данных информационной базы (документов, записей регистров, элементов справочников и др.) ранее определенной даты. Администратор системы может настроить как одну общую дату для всей информационной базы в целом, так и несколько дат по разделам и/или отдельным объектам разделов учета.

Подсистема всегда предоставляет возможность настройки дат запрета как для всех пользователей, так и для конкретных пользователей, групп пользователей (опционально – для внешних пользователей, групп внешних пользователей).

Интерфейс настройки разработан от минимальных возможностей «нет запрета» до максимальных возможностей, определенных при внедрении, но не более чем **По разделам и объектам** в сочетании с настройкой **По пользователям** (и группам пользователей).

Подсистема позволяет устанавливать как произвольные даты запрета, так и относительные даты запрета (конец прошлого месяца, конец прошлого квартала и другие).

Настройка

1. Определить состав типов, для которых будет выполняться запрет изменения данных.
2. Определить состав типов, для которых будут настраиваться запреты загрузки данных.
3. Определить максимальные возможности настройки дат запрета (общая дата, по разделам, по объектам, по разделам и объектам).
4. Определить поля объектов метаданных, которые будут использованы для получения проверяемых данных (даты, раздела, объекта).
5. Определить случаи условного отказа от проверки запрета изменения и определить случаи проверки запрета загрузки.
6. Определить возможность настройки дат запрета для внешних пользователей и групп внешних пользователей.

Определить состав типов, для которых будет выполняться запрет изменения данных

Определить состав типов справочников, документов, регистров, для которых будет выполняться запрет изменения данных.

Создать подписки на события и указать в них выбранные типы согласно таблице, приведенной ниже.

Подписки на событие `ПередЗаписью` для проверки запрета изменения данных:

Имя	Источник	
`ПроверитьДатуЗапретаИзмененияПередЗаписьюДокумента`	Элементы типа `ДокументОбъект`.	`ДатыЗапретаИзменения.ПроверитьДатуЗа`
`ПроверитьДатуЗапретаИзмененияПередЗаписью`	Элементы типов: `СправочникОбъект`, `ПланВидовХарактеристикОбъект`, `ПланСчетовОбъект`, `ПланВидовРасчетаОбъект`, `БизнесПроцессОбъект`, `ЗадачаОбъект`, `ПланОбменаОбъект`.	`ДатыЗапретаИзменения. ПроверитьДатуЗ`
`ПроверитьДатуЗапретаИзмененияПередЗаписьюНабораЗаписей`	Элементы типов: `РегистрСведенийНаборЗаписей`, `РегистрНакопленияНаборЗаписей`.	`ДатыЗапретаИзменения. ПроверитьДатуЗ`
`ПроверитьДатуЗапретаИзмененияПередЗаписьюНабораЗаписейРегистраБухгалтерии`	Элементы типа `РегистрБухгалтерииНаборЗаписей`.	`ДатыЗапретаИзменения. ПроверитьДатуЗ`
`ПроверитьДатуЗапретаИзмененияПередЗаписьюНабораЗаписейРегистраРасчета`	Элементы типа `РегистрРасчетаНаборЗаписей`.	`ДатыЗапретаИзменения. ПроверитьДатуЗ`

Подписки на событие `ПередУдалением` для проверки запрета изменения данных:

Имя	Источник	Обработчик
`ПроверитьДатуЗапретаИзмененияПередУдалением`	Элементы типов: `СправочникОбъект`, `ДокументОбъект`, `ПланВидовХарактеристикОбъект`, `ПланСчетовОбъект`, `ПланВидовРасчетаОбъект`, `БизнесПроцессОбъект`, `ЗадачаОбъект`, `ПланОбменаОбъект`.	`ДатыЗапретаИзменения. ПроверитьДатуЗапретаИзмененияПередУдалением`

Для выбранных объектов метаданных необходимо вставить в модуль формы объекта в процедуру обработчика события `ПриЧтенииНаСервере` вызов:

```
ДатыЗапретаИзменения.ОбъектПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект).
```

При выполнении вызова будет выполнена проверка запрета изменения, и при наличии запрета свойство формы `ТолькоПросмотр` будет установлено в `Истина`.

Определить состав типов, для которых будут настраиваться запреты загрузки данных

Для запрета загрузки данных по узлам планов обмена необходимо в определяемый тип `АдресатЗапретаИзменения` добавить типы требуемых планов обмена. Если хотя бы один тип плана обмена добавлен в состав определяемого типа `АдресатЗапретаИзменения`, то по команде **Даты запрета загрузки** данных будет открыта форма **Даты запрета изменения данных** в режиме редактирования дат запрета загрузки.

Определить максимальные возможности настройки дат запрета (общая дата, по разделам, по объектам, по разделам и объектам)

Для различных прикладных решений требования к настройке даты запрета сильно отличаются. Для простой однопользовательской конфигурации достаточно одной общей даты, а для корпоративного решения требуется множество разрезов дат запрета к различным областям планирования и учета. Для этих целей предусмотрены разделы – это разрезы областей планирования и учета для установки дат запрета. Например, **Складской учет**, **Бухгалтерская отчетность**, **Планирование**. В подсистеме разделы – это элементы объекта метаданных `ПланВидовХарактеристик.РазделыДатЗапретаИзменения`, которые создаются автоматически на основе описания разделов в процедуре `ПриЗаполненииРазделовДатЗапретаИзменения` общего модуля `ДатыЗапретаИзмененияПереопределяемый` (режиме **1С:Предприятие** элементы не редактируются).

При необходимости разделы могут быть уточнены объектом – например, раздел **Складской учет** можно уточнить складом. Для этого разделу нужно установить тип, например `СправочникСсылка.Склады`.

Возможны четыре варианта внедрения:

1. **Общая дата**. В этом варианте список разделов пуст, у пользователя будет возможность задавать общую дату на всю конфигурацию (для всех пользователей или по пользователям).
2. **По разделам**. В этом варианте список разделов состоит из двух и более разделов, типы разделов не определены (`ПланВидовХарактеристикСсылка.РазделыДатЗапретаИзменения`). В интерфейсе объекты не отображаются, их нельзя добавить. Например, разделы **Учет** и **Планирование**. При этом можно задать общую дату или две даты отдельно по каждому разделу.
3. **По объектам**. В этом варианте список разделов состоит из одного раздела. Этот раздел не отображается в интерфейсе, но используется при получении данных для проверки. У раздела задан непустой тип объектов. Например, раздел **Бухгалтерский учет** с типом

`СправочникСсылка.Организации`. При этом можно задать общую дату или даты по отдельным организациям.

4. **По разделам и объектам**. В этом самом общем варианте список разделов состоит из двух и более разделов, причем хотя бы один раздел должен иметь тип объектов. Например, **Учет по организациям** и **Планирование**.

```
// Заполняет разделы дат запрета изменения, используемые при настройке дат запрета.
// Если не указать ни одного раздела, тогда будет доступна только настройка общей даты запрета.
//
// Параметры:
//  Разделы - ТаблицаЗначений - с колонками:
//   * Имя - Строка - имя используемое в описании источников данных в
//       процедуре ЗаполнитьИсточникиДанныхДляПроверкиЗапретаИзменения.
//
//   * Идентификатор - УникальныйИдентификатор - идентификатор ссылки элемента плана видов характеристик.
//       Чтобы получить идентификатор нужно в режиме 1С:Предприятия выполнить метод платформы:
//       "ПланыВидовХарактеристик.РазделыДатЗапретаИзменения.ПолучитьСсылку().УникальныйИдентификатор()".
//       Не следует указывать идентификаторы, полученные любым другим способом,
//       так как это может нарушить их уникальность.
//
//   * Представление - Строка - представляет раздел в форме настройки дат запрета.
//
//   * ТипыОбъектов  - Массив - типы ссылок объектов в разрезе которых можно настроить даты запрета,
//       например, Тип("СправочникСсылка.Организации"), если не указано ни одного типа,
//       то даты запрета будут настраиваться только с точностью до раздела.
//
Процедура ПриЗаполненииРазделовДатЗапретаИзменения(Разделы) Экспорт

    Раздел = Разделы.Добавить();
    Раздел.Имя  = "_ДемоБанк";
    Раздел.Идентификатор = Новый УникальныйИдентификатор("4109a54a-f3ea-474c-9079-be08bf335668");
    Раздел.Представление = НСтр("ru = 'Демо: Банк'");
    Раздел.ТипыОбъектов.Добавить(Тип("СправочникСсылка._ДемоБанковскиеСчета"));
    Раздел = Разделы.Добавить();
    Раздел.Имя  = "_ДемоНачислениеЗарплаты";
    Раздел.Идентификатор = Новый УникальныйИдентификатор("100aba96-ea50-4f82-a06c-2e3fdc39a9f1");
    Раздел.Представление = НСтр("ru = 'Демо: Начисление зарплаты'");

КонецПроцедуры
```

Определить поля объектов метаданных, которые будут использованы для получения проверяемых данных (даты, раздела, объекта)

Для подготовки данных к проверке на запрет изменения требуется для каждого из выбранных объектов метаданных определить поля – источники значений. Для этих целей используется процедура `ЗаполнитьИсточникиДанныхДляПроверкиЗапретаИзменения` переопределяемого модуля.

```
// Содержит описание таблиц и полей объектов для проверки запретов изменения данных.
//   Вызывается из процедуры ИзменениеЗапрещено общего модуля ДатыЗапретаИзменения,
//   используемой в подписке на событие ПередЗаписью объекта для проверки наличия
//   запретов и отказа от изменений запрещенного объекта.
//
// Параметры:
//  ИсточникиДанных - ТаблицаЗначений - с колонками:
//   * Таблица     - Строка - полное имя объекта метаданных,
//                   например Метаданные.Документы.ПриходнаяНакладная.ПолноеИмя().
//   * ПолеДаты    - Строка - имя реквизита объекта или табличной части,
//                   например, "Дата", "Товары.ДатаОтгрузки".
//   * Раздел      - Строка - имя раздела дат запрета, указанного в
//                   процедуре ПриЗаполненииРазделовДатЗапретаИзменения (см. выше).
//   * ПолеОбъекта - Строка - имя реквизита объекта или реквизита табличной части,
//                   например, "Организация", "Товары.Склад".
//
//  Для добавления строки имеется процедура ДобавитьСтроку в общем модуле ДатыЗапретаИзменения.
//
Процедура ЗаполнитьИсточникиДанныхДляПроверкиЗапретаИзменения(ИсточникиДанных) Экспорт
    ДатыЗапретаИзменения.ДобавитьСтроку(ИсточникиДанных,
        Метаданные.Документы._ДемоОприходованиеТоваров.ПолноеИмя(),
        "Дата", "_ДемоСкладскойУчет", "МестоХранения");

    ДатыЗапретаИзменения.ДобавитьСтроку(ИсточникиДанных,
        Метаданные.Документы._ДемоСписаниеТоваров.ПолноеИмя(),
        "Дата", "_ДемоСкладскойУчет", "МестоХранения");

    ДатыЗапретаИзменения.ДобавитьСтроку(ИсточникиДанных,
        Метаданные.Документы._ДемоСписаниеБезналичныхДенежныхСредств.ПолноеИмя(),
        "ДатаПроведенияБанком", "_ДемоБанк", "БанковскийСчет");

КонецПроцедуры
```

Поле может быть задано «через точку» – например, если в документе указана ссылка на документ-основание, содержащий объект. Тогда реквизит будет задан так: `ДокументОснование.Склад`, где `ДокументОснование` – реквизит документа, проверяемого на запрет изменения. Можно

указать поле «через несколько точек».

В качестве поля можно указать поле табличной части, в этом случае используется тот же метод «через точку», например **Товары.Склад**. Указывать поле табличной части можно только в проверяемом документе, а не в документе-основании (такое описание не поддерживается).

Следует иметь в виду, что использование метода «через точку» потребует дополнительного обращения к базе данных для получения значения, а использование поля табличной части увеличит количество проверяемых данных.

Для объекта данных может потребоваться несколько проверок (например, для документа `ПеремещениеТоваров` требуется проверить запрет изменения по двум складам). В этом случае используются несколько строк для описания проверок по одному объекту метаданных (см. пример выше).

Определить случаи условного отказа от проверки запрета изменения и определить случаи проверки запрета загрузки

Кроме стандартного случая отказа от проверки запрета изменения и запрета загрузки данных при обновлении информационной базы, прикладной разработчик может пропустить проверку запретов по прикладным причинам.

Если в прикладной конфигурации потребовалось пропустить проверку запрета изменения данных – например, если заказ не закрыт (или при особых случаях обмена данными пропустить проверку запрета загрузки данных), следует воспользоваться переопределяемой процедурой `ПередПроверкойЗапретаИзменения`.

```
// Позволяет переопределить выполнение проверок запретов по произвольному условию.
//
// Параметры:
//  Объект       - СправочникОбъект,
//                 ДокументОбъект,
//                 ПланВидовХарактеристикОбъект,
//                 ПланСчетовОбъект,
//                 ПланВидовРасчетаОбъект,
//                 БизнесПроцессОбъект,
//                 ЗадачаОбъект,
//                 ПланОбменаОбъект - объект данных (ПередЗаписью или ПриЧтенииНаСервере).
//               - РегистрСведенийНаборЗаписей,
//                 РегистрНакопленияНаборЗаписей,
//                 РегистрБухгалтерииНаборЗаписей,
//                 РегистрРасчетаНаборЗаписей - набор записей (ПередЗаписью или ПриЧтенииНаСервере).
//
//  ПроверкаЗапретаИзменения    - Булево - когда Истина, тогда проверка изменения выполняется.
//                                Если установить Ложь, тогда проверка запрета изменения будет пропущена.
//
//  УзелПроверкиЗапретаЗагрузки - Неопределено - проверка запрета загрузки не выполняется.
//                              - ПланыОбменаСсылка - проверка запрета загрузки выполняется. Если
//                                установить Неопределено, проверка запрета загрузки будет пропущена.
//
//  ВерсияОбъекта               - Строка - начальное значение "". Проверяются обе версии объекта.
//                                Если установить "СтараяВерсия" или "НоваяВерсия", тогда будет
//                                выполнена проверка только старой или только новой версии объекта.
//
Процедура ПередПроверкойЗапретаИзменения(Объект,
                                         ПроверкаЗапретаИзменения,
                                         УзелПроверкиЗапретаЗагрузки,
                                         ВерсияОбъекта) Экспорт

    Если ТипЗнч(Объект) = Тип("ДокументОбъект._ДемоЗаказПокупателя") Тогда
        СтатусЗаказа = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Объект.Ссылка, "СтатусЗаказа");
        ЗаказЗакрытСтараяВерсия = (СтатусЗаказа        = Перечисления._ДемоСтатусыЗаказовПокупателей.Закрыт);
        ЗаказЗакрытНоваяВерсия  = (Объект.СтатусЗаказа = Перечисления._ДемоСтатусыЗаказовПокупателей.Закрыт);

        Если Не ЗаказЗакрытСтараяВерсия И Не ЗаказЗакрытНоваяВерсия Тогда
            ПроверкаЗапретаИзменения = Ложь;
            УзелПроверкиЗапретаЗагрузки = Неопределено;

        ИначеЕсли Не ЗаказЗакрытНоваяВерсия Тогда
            ВерсияОбъекта = "СтараяВерсия"; // Проверить только старую версию объекта.

        ИначеЕсли Не ЗаказЗакрытСтараяВерсия Тогда
            ВерсияОбъекта = "НоваяВерсия"; // Проверить только новую версию объекта.
        КонецЕсли;

    ИначеЕсли ТипЗнч(Объект) = Тип("ДокументОбъект._ДемоСписаниеБезналичныхДенежныхСредств") Тогда
        // Отказ от проверки с учетом того, что ДатаПроведенияБанком, используемая в проверке не указывается,
        // если документ не проведен банком, а указывается позже после проведения документа банком.
        ПроведеноБанкомНоваяВерсия  = Объект.ПроведеноБанком;
        ПроведеноБанкомСтараяВерсия = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(
            Объект.Ссылка, "ПроведеноБанком") = Истина;

        Если Не ПроведеноБанкомНоваяВерсия И Не ПроведеноБанкомСтараяВерсия Тогда
            ПроверкаЗапретаИзменения = Ложь;
            УзелПроверкиЗапретаЗагрузки = Неопределено;

        ИначеЕсли Не ПроведеноБанкомНоваяВерсия Тогда
            ВерсияОбъекта = "СтараяВерсия"; // Проверить только старую версию объекта.

        ИначеЕсли Не ПроведеноБанкомСтараяВерсия Тогда
            ВерсияОбъекта = "НоваяВерсия"; // Проверить только новую версию объекта.
        КонецЕсли;
    КонецЕсли;

КонецПроцедуры
```

Определить возможность настройки дат запрета для внешних пользователей и групп внешних пользователей

Если в прикладном решении требуется делать настройку по внешним пользователям и группам внешних пользователей, то нужно вписать следующий код в процедуру `НастройкаИнтерфейса` переопределяемого модуля:

```
Процедура НастройкаИнтерфейса(Знач НастройкиРаботыИнтерфейса) Экспорт

    НастройкиРаботыИнтерфейса.ИспользоватьВнешнихПользователей = Истина;

КонецПроцедуры
```

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, необходимо разместить в командном интерфейсе ответственного за даты запрета объекты метаданных:

- команду `ДатыЗапретаИзмененияДанных` – для просмотра и редактирования дат запрета изменения;
- команду `ДатыЗапретаЗагрузкиДанных` – для просмотра и редактирования дат запрета загрузки (если запреты загрузки не используются, команду размещать не требуется).

Для размещения команд требуется включить регистр сведений `ДатыЗапретаИзменения` в подсистему интерфейса (у регистра отключено использование стандартных команд).

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Даты запрета изменения** следует использовать роли:

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Просмотр и редактирование дат запрета изменения и дат запрета загрузки (доступ к форме **Даты запрета изменения данных** с отчетами).
2.	`ЧтениеДатЗапретаИзменения` Просмотр дат запрета изменения (доступ к форме **Даты запрета изменения данных** и отчету **Даты запрета изменения данных**).
3.	`ДобавлениеИзменениеДатЗапретаИзменения` Просмотр и редактирование дат запрета изменения (доступ к форме **Даты запрета изменения данных** и отчету **Даты запрета изменения данных**).
4.	`ЧтениеДатЗапретаЗагрузки` Просмотр дат запрета загрузки (доступ к форме **Даты запрета загрузки данных** и отчету **Даты запрета загрузки данных**). См. также Обмен данными.
5.	`ДобавлениеИзменениеДатЗапретаЗагрузки` Просмотр и редактирование дат запрета загрузки (доступ к форме **Даты запрета загрузки данных** и отчету **Даты запрета загрузки данных**). См. также Обмен данными.

Использование при разработке конфигурации

При создании новых прикладных объектов метаданных нужно повторно выполнять соответствующую часть процедуры настройки, описанной выше.

Для дополнительной проверки запрета изменения (вне подписок на события) используется функция `ИзменениеЗапрещено` модуля `ДатыЗапретаИзменения`.

Также может потребоваться выполнить программную проверку запрета изменения нестандартного вида. Для этого предусмотрена функция `НайденЗапретИзмененияДанных` модуля `ДатыЗапретаИзменения`, которая выполнит проверку и вернет ее результат, а также отправит стандартное сообщение пользователю об ошибке.

Для работы функции нужно подготовить данные, шаблон которых легко получить (см. комментарий функции). Эта функция используется в подписке на событие перед записью. Данные, полученные из объекта данных, передаются в функцию дважды – для старого и нового экземпляров объекта.

```
// Выполняет поиск дат запрета по проверяемым данным
// для авторизованного пользователя или узла плана обмена.
//
// Параметры:
//  ДанныеДляПроверки - ТаблицаЗначений - возвращается функцией ШаблонДанныхДляПроверки
//                      общего модуля ДатыЗапретаИзменения.
//
//  ОписаниеДанных    - Неопределено - текст сообщения о запрете не формируется.
//                    - Структура - со свойствами:
//                      * НоваяВерсия - Булево - если Истина, то сообщение о запрете
//                                      формировать для новой версии, иначе для старой версии.
//                      * Данные - Ссылка,Объект - ссылка или объект данных для получения
//                                      представления, которое будет использовано в сообщении о запрете.
//                               - НаборЗаписей - набор записей регистра для получения
//                                      представления, которое будет использовано в сообщении о запрете.
//                               - Структура - со свойствами для сообщения о запрете:
//                                  * Регистр - Строка - полное имя регистра;
//                                  * Отбор   - Отбор - отбор набора записей.
//                               - Строка - подготовленное представление данных,
//                                 которое будет использовано в сообщении о запрете.
//
//  ОписаниеОшибки    - Строка - (возвращаемое значение) описание запрета изменения или загрузки.
//                      Не заполняется, если параметр ОписаниеДанных не указан.
//                    - Null - в этом случае текст сообщения об ошибке не будет подготовлен,
//                      что ускоряет выполнение проверки, когда запрет найден.
//
//  УзелПроверкиЗапретаЗагрузки - Неопределено - выполнить проверку изменения данных.
//                              - ПланыОбменаСсылка.<Имя плана обмена> - выполнить проверку
//                                загрузки данных для указанного узла.
//
// Возвращаемое значение:
//  Булево - если Истина, значит найден хотя бы один запрет изменения.
//
Функция НайденЗапретИзмененияДанных(Знач ДанныеДляПроверки,
                                    ОписаниеДанных = Неопределено,
                                    ОписаниеОшибки = Null,
                                    УзелПроверкиЗапретаЗагрузки = Неопределено) Экспорт
```

Взаимодействие при обмене данными

Для проверки запрета загрузки следует использовать функцию `ЗагрузкаЗапрещена` модуля `ДатыЗапретаИзменения`.

Если при обмене данными требуется выполнить запись, когда признак `ОбменДанными.Загрузка` не установлен, но без проверки запрета изменения, то в свойства объекта `ДополнительныеСвойства` нужно вставить свойство `ПропуститьПроверкуЗапретаИзменения`. Когда требуется пропустить проверку запрета изменения для всех объектов, которые записываются при записи целевого объекта, следует вместо вставки свойства `ПропуститьПроверкуЗапретаИзменения` использовать процедуру `ОтключитьПроверкуДатЗапрета` общего модуля `ДатыЗапретаИзменения`.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме следующих:

- константа `ВерсияДатЗапретаИзменения`,
- план видов характеристик `РазделыДатЗапретаИзменения`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется исключать все объекты метаданных подсистемы.

Дополнительные отчеты и обработки

Подсистема **Дополнительные отчеты и обработки** предназначена для подключения и использования дополнительных (внешних) отчетов и обработок к информационной базе в режиме **1С:Предприятие**.

В зависимости от назначения дополнительные отчеты и обработки бывают глобальными, если используются в целом для конфигурации, либо назначаемыми, если используются с конкретными типами объектов информационной базы. Назначаемые обработки делятся по своей функциональности на четыре типа: для заполнения объекта, печатные формы, для создания на основании и отчеты.

Настройка

Для использования подсистемы в конфигурации необходимо поместить справочник `ДополнительныеОтчетыИОбработки` в командный интерфейс администратора. Затем выполнить шаги, описанные ниже.

Настройка назначаемых дополнительных отчетов и обработок

Необходимо принять решение по поводу состава объектов метаданных, к которым должны подключаться команды дополнительных отчетов и обработок. Например, это могут быть все справочники и документы конфигурации. Эти объекты метаданных необходимо указать в составе свойстве `Тип` определяемого типа `ОбъектСДополнительнымиКомандами`.

Для каждого объекта метаданных необходимо внести изменения во все его формы объекта и списка, встроив подсистему Подключаемые команды.

Состав объектов метаданных, к которым можно подключать дополнительные печатные формы, определяется при внедрении подсистемы Печать.

Настройка глобальных дополнительных отчетов и обработок

1. Необходимо принять решение по поводу места размещения в командном интерфейсе конфигурации команд для вызова глобальных отчетов и обработок. Рекомендуется размещать такие команды в подсистемах верхнего уровня, участвующих в формировании командного интерфейса (разделы глобального командного интерфейса).
2. Затем для каждого места размещения глобальных дополнительных обработок создать общую команду вида `ДополнительныеОбработки<место_размещения>`, включить ее в состав функциональной опции `ИспользоватьДополнительныеОтчетыИОбработки` и добавить в нее код по шаблону:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

    ДополнительныеОтчетыИОбработкиКлиент.ОткрытьФормуКомандДополнительныхОтчетовИОбработок(
        ПараметрКоманды,
        ПараметрыВыполненияКоманды,
        ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиДополнительнаяОбработка(),
        "<Имя раздела>");

КонецПроцедуры
```

   где `<имя раздела>` – имя раздела командного интерфейса, в котором размещена команда. Для получения имени рабочего стола следует использовать функцию `ИдентификаторРабочегоСтола` общего модуля `ДополнительныеОтчетыИОбработкиКлиентСервер`. В качестве примера такой команды см. общую команду `ДополнительныеОбработкиАдминистрирование` в демонстрационной конфигурации.
3. Аналогичным образом для каждого места размещения глобальных дополнительных отчетов надо создать общую команду вида `ДополнительныеОтчеты<место_размещения>`, включить ее в состав функциональной опции `ИспользоватьДополнительныеОтчетыИОбработки` и вставить в нее следующий код:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)

    ДополнительныеОтчетыИОбработкиКлиент.ОткрытьФормуКомандДополнительныхОтчетовИОбработок(
            ПараметрКоманды,
            ПараметрыВыполненияКоманды,
            ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиДополнительныйОтчет(),
            "<Имя раздела>");

КонецПроцедуры
```

   где `<имя раздела>` – имя раздела командного интерфейса, в котором размещена команда. Для получения имени рабочего стола следует использовать функцию `ИдентификаторРабочегоСтола` общего модуля `ДополнительныеОтчетыИОбработкиКлиентСервер`. В качестве примера такой команды см. общую команду `ДополнительныеОтчетыАдминистрирование` в демонстрационной конфигурации.
4. В переопределяемом модуле `ДополнительныеОтчетыИОбработкиПереопределяемый` в процедурах `ОпределитьРазделыСДополнительнымиОбработками` и `ОпределитьРазделыСДополнительнымиОтчетами` перечислить в виде объектов метаданных разделы, в которых были размещены общие команды, созданные на предыдущих шагах. См. пример кода в демонстрационной конфигурации.

Настройка необходимости проведения документов перед формированием внешних печатных форм

По умолчанию перед формированием любой внешней печатной формы проверяется проведенность печатаемых объектов-документов, и если найдется хотя бы один непроведенный документ, то пользователю будет выдано соответствующее предложение его провести. Если пользователь отказывается это делать, печать не выполняется.

Такую проверку можно отключить при внедрении подсистемы. Для этого необходимо в общем модуле `ДополнительныеОтчетыИОбработкиКлиентПереопределяемый` в процедуре `ПередВыполнениемКомандыПечатиВнешнейПечатнойФормы` установить параметру `СтандартнаяОбработка` значение `Ложь`:

```
Процедура ПередВыполнениемКомандыПечатиВнешнейПечатнойФормы(ПечатаемыеОбъекты, СтандартнаяОбработка) Экспорт

    СтандартнаяОбработка = Ложь;

КонецПроцедуры
```

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить:

- константу `ИспользоватьДополнительныеОтчетыИОбработки`. При переключении значения константы регулируется возможность использования в информационной базе механизмов подсистемы. Если конфигурация адаптируется для использования в модели сервиса, отображать

команду в форме настроек команды нужно только в локальном режиме работы – в модели сервиса управление значением константы выполняется централизованно из менеджера сервиса;
- команду открытия списка справочника `ДополнительныеОтчетыИОбработки`.

См. пример в форме `ПечатныеФормыОтчетыИОбработки` обработки `ПанельАдминистрированияБСП`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Дополнительные отчеты и обработки** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`ДобавлениеИзменениеДополнительныхОтчетовИОбработок` Добавление дополнительных обработок и отчетов в приложение, изменение параметров публикации и объектов назначения обработок и отчетов.
2.	`ЧтениеДополнительныхОтчетовИОбработок` Возможность запуска разрешенных дополнительных отчетов и обработок. При совместном использовании с подсистемой **Управление доступом** для ограничения доступа к отдельным дополнительным отчетам и обработкам можно использовать вид доступа `ДополнительныеОтчетыИОбработки`.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным, которые не относятся к подсистеме **Дополнительные отчеты и обработки**, но требуются для работы с ней.

№	Вспомогательные роли и их назначение
1.	**Новая или существующая роль**. Роль с установленным правом просмотра общих команд доступа к глобальным отчетам и обработкам (созданных на шагах 2 и 3 раздела настройки глобальных дополнительных отчетов и обработок). Назначается пользователям вместе с ролью `ЧтениеДополнительныхОтчетовИОбработок`.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор дополнительных отчетов и обработок. Имеет возможность добавлять новые обработки в конфигурацию, публиковать их для общего использования, задавать объекты назначения и использовать внешние обработки в конфигурации.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеДополнительныхОтчетовИОбработок`.
2.	Пользователь с правом выполнения дополнительных отчетов и обработок. Имеет возможность запускать обработки на исполнение.	• `БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеДополнительныхОтчетовИОбработок`, • `ЧтениеГлобальныхДополнительныхОтчетовИОбработок`.

Особые случаи внедрения подсистемы

- Внедрение подсистем «Дополнительные отчеты и обработки» и «Управление доступом».

Использование при разработке конфигурации

Создание нового отчета или обработки

Дополнительные отчеты и обработки могут быть глобальными (виды `ДополнительнаяОбработка` и `ДополнительныйОтчет`) и назначаемыми (виды `ЗаполнениеОбъекта`, `Отчет`, `ПечатнаяФорма` и `СозданиеСвязанныхОбъектов`). Глобальные команды размещаются в глобальном командном интерфейсе. Назначаемые команды используются вместе с объектами информационной базы и размещаются в контексте тех объектов, для которых они предназначаются. Примеры назначаемых обработок см. в справочнике **Демо: Контрагенты** и в документе **Демо: Счета на оплату покупателям** в демонстрационной конфигурации. Общую информацию о работе с внешними отчетами и обработками можно найти в разделе 5.12.2. «Внешние отчеты и обработки» документации к платформе.

В общем виде последовательность создания дополнительной обработки (отчета) следующая:

1. Создать внешнюю обработку (отчет).
2. Регистрация дополнительной обработки в информационной базе происходит на основании сведений, которые поставляет сама обработка. Эти сведения должны возвращаться в виде структуры в функции `СведенияОВнешнейОбработке` модуля обработки. Поля структуры имеют следующее назначение:

Ключ	Содержание
`Вид`	Строка, вид обработки, один из возможных: `ДополнительнаяОбработка`, `ДополнительныйОтчет`, `ЗаполнениеОбъекта`, `Отчет`, `ПечатнаяФорма` или `СозданиеСвязанныхОбъектов`
`Назначение`	Массив строк имен объектов метаданных в формате: `ИмяКлассаОбъектаМетаданного.[*
`Наименование`	Наименование обработки, которым будет заполнено наименование элемента справочника, краткое описание для идентификации обработки администратором
`Версия`	Версия обработки в формате `<старший номер>``<младший номер>` используется при загрузке обработок в информационную базу. Например, **1.0**
`БезопасныйРежим`	Принимает значение `Истина` или `Ложь` в зависимости от того, требуется устанавливать или отключать безопасный режим при исполнении обработки (отчета). Если `Истина`, обработка (отчет) будет запущена в безопасном режиме. При этом, в частности, не гарантируется корректная работа дополнительных обработок под пользователями без полных прав в тех случаях, когда обработка вызывает код конфигурации, который не рассчитан на работу в безопасном режиме (в том числе неявно, когда срабатывает код модуля объекта и подписок на события при записи данных). Более подробно о безопасном режиме см. документацию к платформе **1С:Предприятие**
`Информация`	Краткая информация по обработке, описание обработки
`Команды`	Команды, поставляемые обработкой. Таблица значений с колонками: • `Представление` – представление команды в пользовательском интерфейсе; • `Идентификатор` – идентификатор команды; любая строка, уникальная в пределах данной обработки (отчета); [1] • `Использование` – вариант запуска команды; • `ОткрытиеФормы` – открыть форму обработки; • `ВызовКлиентскогоМетода` – вызвать клиентскую экспортную процедуру из модуля формы обработки; • `ВызовСерверногоМетода` – вызвать серверную экспортную процедуру из модуля объекта обработки; • `СценарийВБезопасномРежиме` – вызвать серверную экспортную функцию из модуля объекта для формирования сценария выполнения (специальный вариант запуска для использования программного интерфейса, расширяющего доступные разработчику дополнительной обработки операции, при выполнении дополнительной обработки в безопасном режиме); • `ЗаполнениеФормы` – из модуля формы объекта вызвать серверную процедуру обработки для заполнения данных формы без записи объекта в базу данных; • `ПоказыватьОповещение` – `Истина`, если требуется показать оповещение при начале и при завершении работы обработки. Например, при запуске обработки без открытия формы; • `Модификатор` – дополнительный модификатор команды. Используется для дополнительных обработок печатных форм на основе табличных макетов. Для таких команд должен содержать строку `ПечатьMXL` (см. пример в демонстрационной конфигурации); • `ЗаменяемыеКоманды` – используется для дополнительных обработок печатных форм, содержит идентификаторы стандартных команд печати (строкой, через запятую), которые заменяются обработкой.
`ВерсияБСП`	Минимальная версия БСП, на которую рассчитывает код дополнительной обработки. Должна быть задана не ниже 1.2.1.4. Номер версии БСП задается в формате «РР.ПП.ВВ.СС» (РР – старший номер редакции; ПП – младший номер редакции; ВВ – номер версии; СС – номер сборки).
`Разрешения`	Массив разрешений, предоставляемых дополнительной обработке при использовании программного интерфейса, расширяющего доступные разработчику дополнительной обработки операции, при выполнении дополнительной обработки в безопасном режиме. Пример использования разрешений см. в обработке `_ДемоЗагрузкаНоменклатурыИзПрайсЛистаПрофилиБезопасности` демонстрационной конфигурации.

`ОпределитьНастройкиФормы`	`Булево`. Используется для дополнительных отчетов, подключенных к подсистеме `Варианты отчетов` и использующих общую форму отчета (подробнее см. шаг 3).
	Когда `Истина`, то дополнительный отчет имеет программный интерфейс для тесной интеграции с формой отчета, в том числе может переопределять некоторые настройки формы и подписываться на ее события. В этом случае в модуле объекта отчета следует определить процедуру по шаблону:
	`// Настройки общей формы отчета подсистемы "Варианты отчетов".`
	`//`
	`// Параметры:`
	`// Форма - ФормаКлиентскогоПриложения, Неопределено - Форма отчета или форма настроек отчета.`
	`// Неопределено, когда вызов без контекста.`
	`// КлючВарианта - Строка, Неопределено - Имя предопределенного`
	`// или уникальный идентификатор пользовательского варианта отчета.`
	`// Неопределено, когда вызов без контекста.`
	`// Настройки - Структура - см. возвращаемое значение`
	`// ОтчетыКлиентСервер.ПолучитьНастройкиОтчетаПоУмолчанию().`
	`// Процедура ОпределитьНастройкиФормы(Форма, КлючВарианта, Настройки) Экспорт`
	`// Код процедуры. КонецПроцедуры`

Ключ	Содержание
`НазначениеВариантаОтчета`	Назначение варианта дополнительного отчета, который будет сформирован при записи дополнительного отчета. Данный параметр имеет смысл только для дополнительных отчетов. Выбор осуществляется из значений перечисления `НазначенияВариантовОтчетов`: `ДляКомпьютеровИПланшетов` - вариант отчета будет отображаться в панелях отчетов при запуске стандартного приложения 1С:Предприятие (если для дополнительного отчета не задан параметр `НазначениеВариантаОтчета`, то варианту отчета будет присвоено данное значение назначение) `ДляСмартфонов` - вариант отчета будет отображаться в панелях отчетов при запуске мобильного приложения 1С:Предприятие `ДляЛюбыхУстройств` - вариант отчета будет отображаться всегда вне зависимости от приложения 1С:Предприятие

[1] Для обработок с печатными формами на основе макета табличного документа должен содержать список макетов, на основе которых нужно получить печатную форму (см. описание параметра `ИменаМакетов` процедуры `УправлениеПечатьюКлиент.ВыполнитьКомандуПечати` в разделе Печать).

Пример реализации функции `СведенияОВнешнейОбработке` с использованием программного интерфейса:

```
Функция СведенияОВнешнейОбработке() Экспорт
    ПараметрыРегистрации = ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.2.2.1");
    ПараметрыРегистрации.Вид = ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработки<...>();
    ПараметрыРегистрации.Версия = "...";

    Команда = ПараметрыРегистрации.Команды.Добавить();
    Команда.Представление = НСтр("ru = '<Представление команды>'");
    Команда.Идентификатор = "<Имя команды>";
    Команда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКоманды<...>();
    Команда.ПоказыватьОповещение = <Истина/Ложь>;

    Возврат ПараметрыРегистрации;
КонецФункции
```

В этом примере также демонстрируется возможность автоматического заполнения параметров `Наименование` и `Информация` из метаданных внешней обработки (берутся синоним и комментарий соответственно). Пример реализации функции `СведенияОВнешнейОбработке` с ручным определением структуры параметров:

```
Функция СведенияОВнешнейОбработке() Экспорт
    ПараметрыРегистрации = Новый Структура;
    ПараметрыРегистрации.Вставить("Вид", ...);
    ПараметрыРегистрации.Вставить("Назначение", ...);
    ПараметрыРегистрации.Вставить("Наименование", ...);
    ПараметрыРегистрации.Вставить("Версия", ...);
    ПараметрыРегистрации.Вставить("Информация", ...);
    ПараметрыРегистрации.Вставить("ВерсияБСП", "1.2.1.4");

    Команды = Новый ТаблицаЗначений;
    Команды.Колонки.Добавить("Представление", Новый ОписаниеТипов("Строка"));
    Команды.Колонки.Добавить("Идентификатор", Новый ОписаниеТипов("Строка"));
    Команды.Колонки.Добавить("Использование", Новый ОписаниеТипов("Строка"));
    Команды.Колонки.Добавить("ПоказыватьОповещение", Новый ОписаниеТипов("Булево"));
    Команды.Колонки.Добавить("Модификатор", Новый ОписаниеТипов("Строка"));
    ПараметрыРегистрации.Вставить("Команды", Команды);

    Команда = ПараметрыРегистрации.Команды.Добавить();
    Команда.Представление = НСтр("ru = '<Представление команды>'");
    Команда.Идентификатор = "<Имя команды>";
    Команда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКоманды<...>();
    Команда.ПоказыватьОповещение = <Истина/Ложь>;
    Возврат ПараметрыРегистрации;
КонецФункции
```

1. Для дополнительных отчетов принять решение по поводу подключения к подсистеме **Варианты отчетов** (если подсистема внедрена в конфигурацию).
2. Подробнее про возможности подсистемы **Варианты отчетов** см. в соответствующем разделе. В частности, варианты дополнительных отчетов, подключенных к подсистеме **Варианты отчетов**, также можно размещать в панелях отчетов приложения.
3. Для подключения отчета к подсистеме **Варианты отчетов** в свойстве **Хранилище вариантов** (меню **Действия**, команда **Свойства**) следует выбрать `ХранилищеВариантовОтчетов`. Если дополнительный отчет подключен к подсистеме **Варианты отчетов** и использует общие формы `ФормаОтчета`, `ФормаНастроекОтчета` и `ФормаВариантаОтчета`, то в функции `СведенияОВнешнейОбработке` появляется возможность подключить обработчики, расширяющие поведение формы отчета. Для этого в свойстве `ПараметрыРегистрации.ОпределитьНастройкиФормы` следует указать значение `Истина` и в модуле объекта отчета определить процедуру по шаблону:

```
// Настройки общей формы отчета подсистемы "Варианты отчетов".
//
// Параметры:
//   Форма - ФормаКлиентскогоПриложения, Неопределено - Форма отчета или форма настроек отчета.
//       Неопределено, когда вызов без контекста.
//   КлючВарианта - Строка, Неопределено - Имя предопределенного
//       или уникальный идентификатор пользовательского варианта отчета.
//       Неопределено, когда вызов без контекста.
//   Настройки - Структура - см. возвращаемое значение
//       ОтчетыКлиентСервер.ПолучитьНастройкиОтчетаПоУмолчанию().
//
Процедура ОпределитьНастройкиФормы(Форма, КлючВарианта, Настройки) Экспорт
    // Код процедуры.
КонецПроцедуры
```

Вариант запуска «Открытие формы»

Для этого варианта запуска необходимо создать форму обработки, которая будет открыта при выполнении команды:

- для глобальных отчетов и обработок не требуется дополнительных действий;
- для назначаемых отчетов и обработок – в форме обработки добавить параметр `ОбъектыНазначения` типа `Произвольный`. В этот параметр передается массив ссылок на объекты, для которых выполняется дополнительная обработка.

> **Примечание**
>
> При использовании варианта запуска `Открытие формы` требуется самостоятельно вызвать метод `ОповеститьОбИзменении` после изменения объекта(ов), для того чтобы отразить изменения в пользовательском интерфейсе.

В форму передаются два параметра: `ИдентификаторКоманды` и `ДополнительнаяОбработкаСсылка` (ссылка на элемент справочника `ДополнительныеОтчетыИОбработки`, который связан с данной дополнительной обработкой).

Вариант запуска «Вызов клиентского метода»

Для этого варианта запуска необходимо создать форму обработки и в модуле формы обработки завести экспортную процедуру определенного вида.

Для глобальных отчетов и обработок – реализовать экспортную процедуру `ВыполнитьКоманду` с параметром `ИдентификаторКоманды` (`Строка`):

```
&НаКлиенте
Процедура ВыполнитьКоманду(ИдентификаторКоманды) Экспорт

  // Реализация логики команды
  // ...

КонецПроцедуры
```

Для назначаемых обработок типа `ПечатнаяФорма` реализовать экспортную процедуру `Печать` с двумя параметрами: `ИдентификаторКоманды` и `ОбъектыНазначенияМассив`, где `ИдентификаторКоманды` – строка, идентификатор команды; `ОбъектыНазначенияМассив` – массив ссылок на объекты информационной базы, для которых выполняется дополнительная обработка:

```
&НаКлиенте
Процедура Печать(ИдентификаторКоманды, ОбъектыНазначенияМассив) Экспорт

  // Реализация логики команды печати
  // ...

КонецПроцедуры
```

Для назначаемых обработок типа **Создание связанных объектов** – реализовать экспортную процедуру `ВыполнитьКоманду` с параметрами `ИдентификаторКоманды`, `ОбъектыНазначенияМассив` и `СозданныеОбъекты`, где `СозданныеОбъекты` – массив ссылок на созданные объекты:

```
&НаКлиенте
Процедура ВыполнитьКоманду(ИдентификаторКоманды, ОбъектыНазначенияМассив, СозданныеОбъекты) Экспорт

  // Реализация логики команды по созданию связанных объектов
  // ...

КонецПроцедуры
```

Для назначаемых обработок типа **Заполнение объекта** и **Отчет** реализовать экспортную процедуру `ВыполнитьКоманду` с параметрами `ИдентификаторКоманды`, `ОбъектыНазначенияМассив`:

```
&НаКлиенте
Процедура ВыполнитьКоманду(ИдентификаторКоманды, ОбъектыНазначенияМассив) Экспорт

  // Реализация логики команды по заполнению объекта
  // ...

КонецПроцедуры
```

Для всех типов дополнительных отчетов и обработок в форму передается параметр `ДополнительнаяОбработкаСсылка` – ссылка на элемент справочника `ДополнительныеОтчетыИОбработки`, который связан с данной дополнительной обработкой или отчетом.

Вариант запуска «Вызов серверного метода»

Для этого варианта запуска необходимо в модуле обработки требуется завести экспортную процедуру определенного вида.

Для глобальных отчетов и глобальных обработок реализовать экспортную процедуру `ВыполнитьКоманду` с параметрами `ИдентификаторКоманды` и `ПараметрыВыполненияКоманд` (подробнее про состав параметров см. комментарий к функции `ТипКомандыВызовСерверногоМетода` модуля `ДополнительныеОтчетыИОбработкиКлиентСервер`):

```
Процедура ВыполнитьКоманду(ИдентификаторКоманды, ПараметрыВыполненияКоманды) Экспорт

  // Реализация логики команды
  Если ИдентификаторКоманды = ... Тогда
    ...
  ИначеЕсли ...

КонецПроцедуры
```

Для назначаемых обработок типа **Создание связанных объектов** реализовать экспортную процедуру `ВыполнитьКоманду` с параметрами `ИдентификаторКоманды`, `ОбъектыНазначения`, `СозданныеОбъекты` и `ПараметрыВыполненияКоманды`:

```
Процедура ВыполнитьКоманду(ИдентификаторКоманды, ОбъектыНазначения, СозданныеОбъекты, ПараметрыВыполненияКоманды) Экспорт

  // Реализация логики команды по созданию связанных объектов
  Если ИдентификаторКоманды = ... Тогда
    ...
  ИначеЕсли ...

КонецПроцедуры
```

Для назначаемых обработок типа `ПечатнаяФорма` реализовать экспортную процедуру `Печать` с параметрами `МассивОбъектов`, `КоллекцияПечатныхФорм`, `ОбъектыПечати` и `ПараметрыВывода`. Описание параметров см. в разделе Печать. При этом в структуре `ПараметрыВывода` содержится свойство `ДополнительнаяОбработкаСсылка` (ссылка на элемент справочника `ДополнительныеОтчетыИОбработки`, который связан с данной дополнительной обработкой).

```
Процедура Печать(МассивОбъектов, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт

  // Реализация логики команды печати
  Если ИдентификаторКоманды = ... Тогда
    ...
  ИначеЕсли ...

КонецПроцедуры
```

Пример: внешняя печатная форма

Обработка с одной печатной формой:

```
Функция СведенияОВнешнейОбработке() Экспорт

    ПараметрыРегистрации = ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.3.1.73");
    ПараметрыРегистрации.Вид = ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиПечатнаяФорма();
    ПараметрыРегистрации.Версия = "1.3";

    // Определение объектов, к которым подключается эта обработка.
    ПараметрыРегистрации.Назначение.Добавить("Документ._ДемоСчетНаОплатуПокупателю");

    // Добавление команды печати "Счет на оплату".
    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Счет на оплату (внешняя печатная форма)'");
    НоваяКоманда.Идентификатор = "СчетНаОплату";
    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыВызовСерверногоМетода();
    НоваяКоманда.Модификатор = "ПечатьMXL";
    …
    Возврат ПараметрыРегистрации;

КонецФункции
Процедура Печать(МассивОбъектов, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "СчетНаОплату");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
    КонецЕсли;

КонецПроцедуры
Функция СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    ....
    Возврат ТабличныйДокумент;
КонецФункции
```

Для подмены встроенной команды печати необходимо указать ее идентификатор в параметре `ЗаменяемыеКоманды`:

```
// Добавление команды печати "Счет на оплату".
НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
…
НоваяКоманда.ЗаменяемыеКоманды = "СчетЗаказ";
```

Список имеющихся команд печати с идентификаторами находится, как правило, в процедуре `ДобавитьКомандыПечати` модуля менеджера объекта. Если подмену делать не надо, параметр можно не указывать.

Обработка может содержать несколько печатных форм:

```
Функция СведенияОВнешнейОбработке() Экспорт

    ПараметрыРегистрации = ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.3.1.73");
    ПараметрыРегистрации.Вид = ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиПечатнаяФорма();
    ПараметрыРегистрации.Версия = "1.3";

    // Определение объектов, к которым подключается эта обработка.
    ПараметрыРегистрации.Назначение.Добавить("Документ._ДемоСчетНаОплатуПокупателю");

    // Добавление команды печати "Счет на оплату".
    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Счет на оплату (внешняя печатная форма)'");
    НоваяКоманда.Идентификатор = "СчетНаОплату";
    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыВызовСерверногоМетода();
    НоваяКоманда.Модификатор = "ПечатьMXL";
    НоваяКоманда.ЗаменяемыеКоманды = "СчетЗаказ";

    // Добавление команды печати "Заказ покупателя".
    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Заказ покупателя (внешняя печатная форма)'");
    НоваяКоманда.Идентификатор = "ЗаказПокупателя";
    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыВызовСерверногоМетода();
    НоваяКоманда.Модификатор = "ПечатьMXL";

    Возврат ПараметрыРегистрации;

КонецФункции
Процедура Печать(МассивОбъектов, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "СчетНаОплату");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
    КонецЕсли;

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "ЗаказПокупателя");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьЗаказПокупателя(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Заказ покупателя (внешняя печатная форма)'");
    КонецЕсли;

КонецПроцедуры
Функция СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    ....
    Возврат ТабличныйДокумент;
КонецФункции
Функция СформироватьЗаказПокупателя(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    ....
    Возврат ТабличныйДокумент;
КонецФункции
```

Обработка может содержать команду печати комплекта печатных форм. Описание команды печати комплекта отличается от команды печати одной печатной формы тем, что параметр `Идентификатор` содержит список идентификаторов печатных форм (этой обработки), входящих в комплект:

```
// Добавление команды печати "Комплект документов".
НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
НоваяКоманда.Представление = НСтр("ru = 'Комплект документов (внешняя печатная форма)'");
НоваяКоманда.Идентификатор = "СчетНаОплату,ЗаказПокупателя,ГарантийноеПисьмо";
НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыВызовСерверногоМетода();
НоваяКоманда.Модификатор = "ПечатьMXL";
```

Пример обработки с командой печати комплекта:

```
Функция СведенияОВнешнейОбработке() Экспорт

    ПараметрыРегистрации = ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.3.1.73");
    ПараметрыРегистрации.Вид = ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиПечатнаяФорма();
    ПараметрыРегистрации.Версия = "1.3";

    // Определение объектов, к которым подключается эта обработка.
    ПараметрыРегистрации.Назначение.Добавить("Документ._ДемоСчетНаОплатуПокупателю");

    // Добавление команды печати "Комплект документов".
    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Комплект документов (внешняя печатная форма)'");
    НоваяКоманда.Идентификатор = "СчетНаОплату,ЗаказПокупателя,ГарантийноеПисьмо";

    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыВызовСерверногоМетода();
    НоваяКоманда.Модификатор = "ПечатьMXL";

    Возврат ПараметрыРегистрации;
    …
КонецФункции
Процедура Печать(МассивОбъектов, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "СчетНаОплату");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
    КонецЕсли;

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "ЗаказПокупателя");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьЗаказПокупателя(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Заказ покупателя (внешняя печатная форма)'");
    КонецЕсли;

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "ГарантийноеПисьмо");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = СформироватьГарантийноеПисьмо(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Гарантийное письмо (внешняя печатная форма)'");
    КонецЕсли;

КонецПроцедуры
Функция СформироватьСчетНаОплатуПокупателю(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    …
    Возврат ТабличныйДокумент;
КонецФункции
Функция СформироватьЗаказПокупателя(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    …
    Возврат ТабличныйДокумент;
КонецФункции
Функция СформироватьГарантийноеПисьмо(МассивОбъектов, ОбъектыПечати)
    ТабличныйДокумент = Новый ТабличныйДокумент;
    …
    Возврат ТабличныйДокумент;
КонецФункции
```

Для назначаемых обработок типа **Заполнение объекта** и других типов обработок необходимо реализовать экспортную процедуру `ВыполнитьКоманду` с параметрами `ИдентификаторКоманды`, `ОбъектыНазначения` и `ПараметрыВыполненияКоманды`:

```
Процедура ВыполнитьКоманду(ИдентификаторКоманды, ОбъектыНазначения, ПараметрыВыполненияКоманды) Экспорт

  // Реализация логики команды по заполнению объекта
  Если ИдентификаторКоманды = ... Тогда
    ...
  ИначеЕсли
    ...
  КонецЕсли;
  …

КонецПроцедуры
```

Пример: обработка заполнения данных формы без записи объекта

Для типа обработок `ЗаполнениеОбъекта` предусмотрена возможность работать напрямую с данными формы (а не со ссылкой на объект), что позволяет не выполнять принудительную запись объекта в форме ни до, ни после выполнения обработки.

Пример команды заполнения данных формы:

```
Функция СведенияОВнешнейОбработке() Экспорт

    ПараметрыРегистрации =ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.2.2.1");

    ПараметрыРегистрации.Вид =ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиЗаполнениеОбъекта();
    ПараметрыРегистрации.Версия = "1.2";
    ПараметрыРегистрации.Назначение.Добавить("Справочник._ДемоКонтрагенты");

    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Заполнить реквизит ""ИНН"" не записывая объект (заполнение формы)'");
    НоваяКоманда.Идентификатор = "ЗаполнитьИНН";
    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыЗаполнениеФормы();

    Возврат ПараметрыРегистрации;
КонецФункции
Процедура ВыполнитьКоманду(ИмяКоманды, ОбъектыНазначения, ПараметрыВыполнения) Экспорт
    Если ИмяКоманды = "ЗаполнитьИНН" Тогда
        Генератор = Новый ГенераторСлучайныхЧисел;

        ЭтаФорма = ПараметрыВыполнения.ЭтаФорма;
        ЭтаФорма.Объект.ИНН = Формат(Генератор.СлучайноеЧисло(1, 999999999), "ЧЦ=12; ЧДЦ=0; ЧВН=; ЧГ=");

        ЭтаФорма.Модифицированность = Истина;

        Сообщение = Новый СообщениеПользователю();
        Сообщение.Поле = "Объект.ИНН";
        Сообщение.Текст = НСтр("ru = 'Поле ""ИНН"" успешно заполнено'");
        Сообщение.Сообщить();
КонецПроцедуры
```

Подробнее см. дополнительную обработку **Демо: Обработка заполнения** в справочнике **Дополнительные отчеты и обработки** демонстрационной конфигурации.

Пример: сохраняемые параметры дополнительной обработки

При необходимости задать параметры для дополнительной обработки можно воспользоваться реквизитом `ХранилищеНастроек` справочника `ДополнительныеОтчетыИОбработки`. Например, для заполнения документа **Распределение прочих затрат** необходимо получать цены по определенному, специально созданному для этих целей виду цен. Этот вид цен можно задавать с помощью сохраняемого параметра. В общем виде в реквизите `ХранилищеНастроек` можно сохранять произвольные параметры дополнительной обработки (например, в виде структуры). Для доступа к реквизиту `ХранилищеНастроек` в команду выполнения обработки передается ссылка на связанный с ней элемент справочника `ДополнительныеОтчетыИОбработки`, читать и записывать настройки необходимо при помощи командного интерфейса подсистемы.

Пример кода для чтения параметра:

```
НастройкиСтруктура = ДополнительныеОтчетыИОбработки.ЗагрузитьНастройки(Параметры.ДополнительнаяОбработкаСсылка);
```

Для записи:

```
ДополнительныеОтчетыИОбработки.СохранитьНастройки(ДополнительнаяОбработкаСсылка, НастройкиСтруктура);
```

Подробнее см. дополнительную обработку **Демо: Загрузка номенклатуры из прайс-листа** в справочнике **Дополнительные отчеты и обработки** демонстрационной конфигурации.

Фоновое выполнение длительных операций

Для того чтобы длительные операции дополнительного объекта запускались в фоновом режиме, необходимо запустить выполнение команды в фоновом задании и после его завершения принять результат.

Программный интерфейс, помогающий запускать длительные операции в фоне, представлен следующими процедурами:

1. В общем модуле `ДополнительныеОтчетыИОбработкиКлиент`: процедуры `ВыполнитьКомандуВФоне` и `ПараметрыВыполненияКомандыВФоне`.
2. В общем модуле `ДополнительныеОтчетыИОбработки`: `ВыполнитьКомандуИзФормыВнешнегоОбъекта` и `ВыполнитьКоманду`.

Рассмотрим подключение по шагам.

Шаг 1. В форме дополнительной обработки добавить ключевые параметры:

- `ДополнительнаяОбработкаСсылка` типа `СправочникСсылка.ДополнительныеОтчетыИОбработки` для хранения ссылки внешнего объекта;
- `ИдентификаторКоманды` типа `Строка` для хранения идентификатора выполняемой команды.

Шаг 2. Запустить длительную операцию при помощи кода:

```
ПараметрыКоманды = ДополнительныеОтчетыИОбработкиКлиент.ПараметрыВыполненияКомандыВФоне(Параметры.ДополнительнаяОбработкаСсылка);
ПараметрыКоманды.СопровождающийТекст = НСтр("ru = '<Вставить представление выполняемой команды...>'");
Обработчик = Новый ОписаниеОповещения("ПослеЗавершенияДлительнойОперации", ЭтотОбъект, СопровождающийТекст);
ДополнительныеОтчетыИОбработкиКлиент.ВыполнитьКомандуВФоне(Параметры.ИдентификаторКоманды, ПараметрыКоманды, Обработчик);
```

Шаг 3. Добавить процедуру для приемки результата:

```
// Параметры:
//  Результат - см. ДлительныеОперацииКлиент.НовыйРезультатДлительнойОперации
//  СопровождающийТекст - Строка
//
&НаКлиенте
Процедура ПослеЗавершенияДлительнойОперации(Результат, СопровождающийТекст) Экспорт

    Если Результат = Неопределено Тогда // Пользователь отменил задание.
        Возврат;
    КонецЕсли;

    Если Результат.Статус = "Ошибка" Тогда
        СтандартныеПодсистемыКлиент.ВывестиИнформациюОбОшибке(Результат.ИнформацияОбОшибке);
        Возврат;
    КонецЕсли;

    // Обработка результата (Статус = "Выполнено").
    ПоказатьОповещениеПользователя(НСтр("ru = 'Успешное завершение'"), , СопровождающийТекст, БиблиотекаКартинок.Успешно32);

КонецПроцедуры
```

Шаг 4. В модуле объекта добавить обработчик команды **Вызов серверного метода** по шаблону, приведенному в разделе Вариант запуска «Вызов серверного метода».

В демонстрационной базе БСП данная возможность представлена в обработках **Демо: Управление полнотекстовым поиском**, **Демо: Загрузка номенклатуры из прайс-листа (профили безопасности)**, **Демо: Обработка заполнения** и **Демо: Создание связанных объектов** в справочнике **Дополнительные отчеты и обработки**.

Сохранение параметров обработок между сеансами запуска

В тех случаях, когда для выполнения обработки требуется запомнить набор некоторых параметров, рекомендуется воспользоваться реквизитом `ХранилищеНастроек` (тип `ХранилищеЗначения`) справочника `ДополнительныеОтчетыИОбработки`. С его помощью, например, в интерактивных обработках можно сохранять последние введенные пользователем значения, а для обработок, выполняемых по регламентному заданию, позволить администратору задавать значения по умолчанию и различные параметры работы.

Пример сохранения параметров:

```
ДополнительнаяОбработкаОбъект = ДополнительнаяОбработкаСсылка.ПолучитьОбъект();
ДополнительнаяОбработкаОбъект.ХранилищеНастроек = Новый ХранилищеЗначения(СохраняемоеЗначение);
ДополнительнаяОбработкаОбъект.Записать();
```

Пример загрузки ранее сохраненных параметров:

```
ХранилищеНастроек = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(ДополнительнаяОбработкаСсылка, "ХранилищеНастроек");
Настройки = ХранилищеНастроек.Получить();
```

См. пример реализации в форме обработки `_ДемоЗагрузкаНоменклатурыИзПрайсЛистаПрофилиБезопасности` в демонстрационной конфигурации.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Для планов обмена автономного рабочего места (АРМ) в модели сервиса константу `ИспользоватьДополнительныеОтчетыИОбработки` следует включать в состав планов обмена, но не включать в состав подписок на события (выполняющих регистрацию данных при записи). Данное требование объясняется тем, что в автономном рабочем месте использование дополнительных отчетов и обработок настраивается независимо от их использования в экземпляре сервиса и должно передаваться только при создании начального образа.

Завершение работы пользователей

Подсистема **Завершение работы пользователей** позволяет завершать существующие соединения с информационной базой и устанавливать блокировку новых соединений с информационной базой на определенный период времени.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, в командный интерфейс администратора приложения необходимо поместить обработки `БлокировкаРаботыПользователей` и `АктивныеПользователи`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Завершение работы пользователей** следует использовать роль, указанную ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**): • блокировка сеансов и завершение работы пользователей (с помощью формы **Блокировка работы пользователей**), • отключение сеансов пользователей из формы **Активные пользователи**
2.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**): • блокировка запуска регламентных заданий, • настройка параметров администрирования кластера **1С:Предприятия**

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между приложениями. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Загрузка данных из файла

Подсистема **Загрузка данных из файла** предоставляет пользовательский и программный интерфейс для загрузки табличных данных из файлов в произвольные справочники и табличные части документов и справочников. Например, с ее помощью можно предусмотреть перенос нормативно-справочной информации при переходе с других приложений, а также быстрое заполнение табличных частей при вводе документов.

Настройка

Загрузка данных в справочники

Если в конфигурации используется подсистема **Настройки программы**, рекомендуется разместить обработку `ЗагрузкаДанныхИзФайла` на панели администрирования **Перенос данных** в группе **Загрузка данных**. См. пример размещения в демонстрационной конфигурации в форме `ПереносДанных` обработки `_ДемоПанельАдминистрирования`.

В противном случае необходимо разместить обработку `ЗагрузкаДанныхИзФайла` в рабочем месте администратора приложения.

По умолчанию загрузка данных становится доступной для большинства справочников конфигурации. Однако это не всегда целесообразно. Рекомендуется запрещать загрузку данных в справочники:

- содержащие служебную, технологическую или другую информацию, изменение которой может привести к некорректной работе приложения;
- которые имеют собственные механизмы автоматического обновления или заполнения. Например, классификаторы `КлассификаторБанковРФ`, `СтраныМира` и пр.;
- пакетная загрузка в которые небезопасна (например, `ВнешниеПользователи`, `Пользователи`, `ТомаХраненияФайлов`);
- «технологические» справочники, которые не редактируют пользователи (например, `ГруппыИсполнителейЗадач`, `ИдентификаторыОбъектовМетаданных`, `ОчередьЗаданий` и т. д.).

В таких случаях в процедуре `ПриОпределенииСправочниковДляЗагрузкиДанных` общего модуля `ЗагрузкаДанныхИзФайлаПереопределяемый` следует исключать такие справочники:

```
Процедура ПриОпределенииСправочниковДляЗагрузкиДанных(ЗагружаемыеСправочники) Экспорт

    // Загрузка в классификатор валюты запрещена
    СтрокаТаблицы = ЗагружаемыеСправочники.Найти(Метаданные.Справочники.Валюты.ПолноеИмя(), "ПолноеИмя");
    Если СтрокаТаблицы <> Неопределено Тогда
        ЗагружаемыеСправочники.Удалить(СтрокаТаблицы);
    КонецЕсли;

КонецПроцедуры
```

Кроме того, по умолчанию загрузка данных запрещена для справочников с реквизитами типа `ХранилищеЗначений` (например, все справочники с присоединенными файлами или с форматированными документами).

Это ограничение обусловлено тем, что стандартный механизм загрузки не позволяет загружать данные в элементы таких справочников, а загрузка только оставшейся части реквизитов может привести к некорректным результатам.

Однако если известно, что загрузка в такой справочник без заполнения этих элементов не вызовет проблем в работе приложения, то можно разрешить загрузку данных, разместив в процедуре `ПриОпределенииСправочниковДляЗагрузкиДанных` общего модуля `ЗагрузкаДанныхИзФайлаПереопределяемый` следующий код:

```
Процедура ПриОпределенииСправочниковДляЗагрузкиДанных(ЗагружаемыеСправочники) Экспорт

    Сведения = ЗагружаемыеСправочники.Добавить();
    Сведения.ПолноеИмя = Метаданные.Справочники._ДемоНоменклатура.ПолноеИмя();
    Сведения.Представление = Метаданные.Справочники._ДемоНоменклатура.Представление();
    Сведения.ПрикладнаяЗагрузка = Ложь;

КонецПроцедуры
```

Также имеется возможность переопределить стандартный механизм загрузки данных в справочники. Подробнее см. в разделе Особые случаи внедрения подсистемы.

Загрузка данных в табличную часть документа или справочника

Нужно принять решение по поводу табличных частей объектов (документов и справочников), в которых востребована пакетная загрузка данных. Например, это могут быть табличные части `Товары` документов поступления и реализации товаров и услуг, строки которых часто заполняются на основании первичных документов: из таблиц Microsoft Excel или файлов другого формата.

Для объектов, в которые требуется добавить возможность загрузки данных из файла в табличную часть, необходимо произвести следующие действия.

Команда загрузки данных

В форме объекта создать команду и разместить ее на командной панели табличной части. Например, для табличной части `Товары` – команду `ЗагрузитьТоварыИзФайла` с синонимом **Загрузить из файла** Установить свойство **Изменяет сохраняемые данные** в `Истина`. Если в объекте требуется предусмотреть загрузку в несколько табличных частей, то для каждой табличной части создается отдельная команда.

Далее в форме объекта реализовать обработку команды и процедуру добавления данных в табличную часть. Пример реализации для табличной части `Товары` документа `_ДемоСчетНаОплатуПокупателю`:

```
&НаКлиенте
Процедура ЗагрузитьТоварыИзФайла(Команда)

    ПараметрыЗагрузки = ЗагрузкаДанныхИзФайлаКлиент.ПараметрыЗагрузкиДанных();
    ПараметрыЗагрузки.ПолноеИмяТабличнойЧасти = "_ДемоПоступлениеТоваров.Товары";
    ПараметрыЗагрузки.Заголовок = НСтр("ru = 'Загрузка списка товаров из файла'");
    ДополнительныеПараметры = Новый Структура();
    ДополнительныеПараметры.Вставить("Контрагент", Объект.Контрагент);
    ДополнительныеПараметры.Вставить("Организация", Объект.Организация);
    ПараметрыЗагрузки.ДополнительныеПараметры = ДополнительныеПараметры;
    Оповещение = Новый ОписаниеОповещения("ЗагрузитьТоварыИзФайлаЗавершение", ЭтотОбъект);
    ЗагрузкаДанныхИзФайлаКлиент.ПоказатьФормуЗагрузки(ПараметрыЗагрузки, Оповещение);

КонецПроцедуры
&НаКлиенте
Процедура ЗагрузитьТоварыИзФайлаЗавершение(АдресЗагруженныхДанных, ДополнительныеПараметры) Экспорт

    Если АдресЗагруженныхДанных = Неопределено Тогда
        Возврат;
    КонецЕсли;

    ЗагрузитьТоварыИзФайлаНаСервере(АдресЗагруженныхДанных);
КонецПроцедуры
&НаСервере
Процедура ЗагрузитьТоварыИзФайлаНаСервере(АдресЗагруженныхДанных)

    ЗагруженныеДанные = ПолучитьИзВременногоХранилища(АдресЗагруженныхДанных);

    ТоварыДобавлены = Ложь;
    Для каждого СтрокаТаблицы Из ЗагруженныеДанные Цикл

        Если Не ЗначениеЗаполнено(СтрокаТаблицы.Номенклатура) Тогда
            Продолжить;
        КонецЕсли;

        НоваяСтрокаТовары = Объект.Товары.Добавить();
        НоваяСтрокаТовары.Номенклатура = СтрокаТаблицы.Номенклатура;
        НоваяСтрокаТовары.Цена = СтрокаТаблицы.Цена;
        НоваяСтрокаТовары.Количество = СтрокаТаблицы.Количество;
        НоваяСтрокаТовары.ЕдиницаИзмерения = СтрокаТаблицы.ЕдиницаИзмерения;
        ТоварыДобавлены = Истина;
    КонецЦикла;

    Если ТоварыДобавлены Тогда
        Модифицированность = Истина;
    КонецЕсли;

КонецПроцедуры
```

Табличный макет для загрузки данных

Далее при объекте необходимо создать табличный макет с именем `ЗагрузкаИзФайла`. Если в объекте предусмотрена загрузка в несколько табличных частей, то к макету добавляется имя табличной части: `ЗагрузкаИзФайла<ИмяТабличнойЧасти>`.

В колонках первой строки макета перечисляются представления имен колонок загружаемых данных, а в свойствах ячейки `Имя` следует указать имя колонки таблицы значений с загружаемыми данными.

Возможны случаи, когда один реквизит табличной части может соответствовать сразу нескольким колонкам загружаемых данных. Например, значение реквизита `Номенклатура` табличной части может быть заполнено по одной из колонок: **Штрихкод**, **Артикул** или **Наименование**. В таких случаях в свойствах ячейки макета `ПараметрРасшифровки` требуется указать имя реквизита табличной части, с которым связана эта колонка. Таким образом, для колонок **Штрихкод**, **Артикул** и **Наименование** в свойстве ячейки `ПараметрРасшифровки` указывается **Номенклатура**.

В свойстве **Примечание** каждой колонки первой строки макета рекомендуется написать подсказку о назначении и типе этой колонки. Например, в колонке **Артикул** может быть указано примечание: «Артикул товара (не более 20 символов)».

См. пример макета для загрузки данных в документе `_ДемоСчетНаОплатуПокупателю`.

Также имеется возможность динамически создавать шаблон с нужным набором колонок для загрузки данных в табличные части объектов. Подробнее см. в разделе Особые случаи внедрения подсистемы.

Процедуры в модуле менеджера документа

В модуле менеджера документа нужно добавить экспортные процедуры:

- `УстановитьПараметрыЗагрузкиИзФайлаВТЧ`, которая позволяет переопределить параметры загрузки. Например, можно изменить стандартное имя макета, если документ поддерживает загрузку данных в несколько табличных частей.
- `СопоставитьЗагружаемыеДанныеИзФайла` для реализации логики сопоставления загружаемых данных с данными в информационной базе.

```
// Переопределяет параметры загрузки данных из файла.
//
// Параметры:
//  Параметры - Структура:
//    * ИмяМакетаСШаблоном - Строка - наименование макета. Например, "ЗагрузкаИзФайла".
//    * ИмяТабличнойЧасти - Строка - Полное имя табличной части. Например, "Документ._ДемоСчетНаОплатуПокупателю.ТабличнаяЧасть.Товары"
//    * ОбязательныеКолонки - Массив из Строка - наименования обязательных для заполнения колонок.
//    * ТипДанныхКолонки - Соответствие из КлючИЗначение:
//        * Ключ - Строка - имя колонки;
//        * Значение - ОписаниеТипов - тип колонки загружаемых данных.
//    * ДополнительныеПараметры - Структура
//
Процедура УстановитьПараметрыЗагрузкиИзФайлаВТЧ(Параметры) Экспорт

КонецПроцедуры

// Производит сопоставление данных, загружаемых в табличную часть ПолноеИмяТабличнойЧасти,
// с данными в ИБ, и заполняет параметры АдресТаблицыСопоставления и СписокНеоднозначностей.
//
// Параметры:
//   АдресЗагружаемыхДанных    – Строка – адрес временного хранилища с таблицей значений, в которой
//                               находятся загруженные данные из файла. Состав колонок:
//     * Идентификатор – Число – порядковый номер строки;
//     * остальные колонки соответствуют колонкам макета ЗагрузкаИзФайла.
//   АдресТаблицыСопоставления – Строка – адрес временного хранилища с пустой таблицей значений,
//                               являющейся копией табличной части документа,
//                                       которую необходимо заполнить из таблицы АдресЗагружаемыхДанных.
//   СписокНеоднозначностей – ТаблицаЗначений – список неоднозначных значений, для которых в ИБ имеется несколько подходящих вариантов.
//       * Колонка       – Строка – имя колонки, в которой была обнаружена неоднозначность;
//       * Идентификатор – Число  – идентификатор строки, в которой была обнаружена неоднозначность.
//   ПолноеИмяТабличнойЧасти   – Строка – полное имя табличной части, в которую загружаются данные.
//   ДополнительныеПараметры   – ЛюбойТип – Любые дополнительные сведения.
//
Процедура СопоставитьЗагружаемыеДанные(АдресЗагружаемыхДанных, АдресТаблицыСопоставления, СписокНеоднозначностей, ПолноеИмяТабличнойЧасти, Допо
КонецПроцедуры
```

При сопоставлении данных возможны неоднозначные ситуации, когда по значениям колонок исходных (загружаемых) данных находятся сразу несколько подходящих объектов информационной базы и поэтому невозможно однозначно определить, какой именно объект информационной базы должен быть присвоен реквизиту текущей строки табличной части. Для разрешения неоднозначностей требуется реализовать экспортную процедуру `ЗаполнитьСписокНеоднозначностей`:

```
// Возвращает список подходящих объектов ИБ для неоднозначного значения ячейки.
//
// Параметры:
//    ПолноеИмяТабличнойЧасти  – Строка – Полное имя табличной части, в которую загружаются данные.
//    ИмяКолонки               – Строка – Имя колонки, в который возникла неоднозначность.
//    СписокНеоднозначностей    – Массив  – Массив для заполнения с неоднозначными данными.
//    ЗагружаемыеЗначенияСтрока – Строка  – Загружаемые данные, на основании которых возникла неоднозначность.
//    ДополнительныеПараметры   – ЛюбойТип – Любые дополнительные сведения
//
Процедура ЗаполнитьСписокНеоднозначностей(ПолноеИмяТабличнойЧасти, СписокНеоднозначностей, ИмяКолонки, ЗагружаемыеЗначенияСтрока, Дополнительны
КонецПроцедуры
```

См. пример реализации в документе `_ДемоСчетНаОплатуПокупателю`.

Особые случаи внедрения подсистемы

Собственный алгоритм загрузки данных

Если для загрузки элементов справочника недостаточно стандартных средств, можно предусмотреть собственный алгоритм загрузки. Для этого требуется:

1. В справочнике создать табличный макет с именем `ЗагрузкаИзФайла`, в котором определить состав колонок. Для быстрого создания макета рекомендуется запустить конфигурацию и в режиме **1С:Предприятие** выполнить команду по загрузке данных в этот справочник. Затем на шаге ввода данных в таблицу сохранить бланк для заполнения в MXL-файл и использовать в качестве заготовки для макета. Для загрузки контактной информации и дополнительных сведений в справочник в макете следует определить специальные колонки **Контактная информация** и **Дополнительные сведения**, обозначающие место, где будут автоматически добавлены колонки с видами контактной информации объекта и его дополнительными реквизитами и свойствами.
2. В модуле менеджера справочника реализовать три экспортные процедуры: `ОпределитьПараметрыЗагрузкиДанныхИзФайла`, `СопоставитьЗагружаемыеДанныеИзФайла` и `ЗагрузитьИзФайла`. В них реализовать логику сопоставления и загрузки данных в реквизиты справочника.

```
// Устанавливает параметры загрузки данных из файла
//
// Параметры:
//  Параметры - см. ЗагрузкаДанныхИзФайла.ПараметрыЗагрузкиИзФайла
//
Процедура ОпределитьПараметрыЗагрузкиДанныхИзФайла(Параметры) Экспорт
КонецПроцедуры

// Производит сопоставление загружаемых данных с данными в ИБ.
//
// Параметры:
//   ЗагружаемыеДанные - см. ЗагрузкаДанныхИзФайла.ТаблицаСопоставления
//
Процедура СопоставитьЗагружаемыеДанныеИзФайла(ЗагружаемыеДанные) Экспорт
КонецПроцедуры

// Загрузка данных из файла.
//
// Параметры:
//  ЗагружаемыеДанные - см. ЗагрузкаДанныхИзФайла.ОписаниеЗагружаемыхДанныеДляСправочников
//  ПараметрыЗагрузки - см. ЗагрузкаДанныхИзФайла.НастройкиЗагрузкиДанных
//  Отказ - Булево    - отмена загрузки. Например, если данные некорректные.
//                          - Булево       - Отмена загрузки
Процедура ЗагрузитьИзФайла(ЗагружаемыеДанные, ПараметрыЗагрузки, Отказ) Экспорт
КонецПроцедуры
```

См. пример реализации в справочнике `_ДемоНоменклатура` в демонстрационной конфигурации.

Программное создание табличного макета для загрузки данных в табличную часть

В случаях, когда набор колонок в шаблоне не статичен, а определяется динамически (например, зависит от текущих значений реквизитов объекта, статуса документа, значений функциональных опций и т. п.), есть возможность программно определить состав колонок табличного макета.

Для этого можно воспользоваться следующими функциями программного интерфейса:

- `СформироватьОписаниеКолонок` общего модуля `ЗагрузкаДанныхИзФайла`;
- `ОписаниеКолонкиМакета` и `КолонкаМакета` общего модуля `ЗагрузкаДанныхИзФайлаКлиентСервер`.

Пример реализации для табличной части `Аналоги` документа `_ДемоНоменклатура`:

```
&НаКлиенте
Процедура ЗагрузитьТоварыИзФайла(Команда)

    ПараметрыЗагрузки = ЗагрузкаДанныхИзФайлаКлиент.ПараметрыЗагрузкиДанных();
    ПараметрыЗагрузки.ПолноеИмяТабличнойЧасти = "_ДемоНоменклатура.Аналоги";
    ПараметрыЗагрузки.Заголовок = НСтр("ru = 'Загрузка списка аналогов из файла'");

    // Описание колонок для макета загрузки комплектации
    ПараметрыЗагрузки.КолонкиМакета = ОписаниеКолонокМакетаДляЗагрузкиАналогов();

    Оповещение = Новый ОписаниеОповещения("ЗагрузитьАналогиИзФайлаЗавершение", ЭтотОбъект);
    ЗагрузкаДанныхИзФайлаКлиент.ПоказатьФормуЗагрузки(ПараметрыЗагрузки, Оповещение);

КонецПроцедуры
&НаСервере
Функция ОписаниеКолонокМакетаДляЗагрузкиАналогов();

    КолонкиМакета = ЗагрузкаДанныхИзФайла.СформироватьОписаниеКолонок(Объект.Аналоги);
    ЗагрузкаДанныхИзФайлаКлиентСервер.УдалитьКолонкуМакета("Аналог", КолонкиМакета);

    Если Объект.ВидНоменклатуры.Наименование = "Услуга" Тогда
        // У услуг нет штрихкода
        Колонка = ЗагрузкаДанныхИзФайлаКлиентСервер.ОписаниеКолонкиМакета("ШтрихкодАртикул", ОбщегоНазначения.ОписаниеТипаСтрока(20), НСтр("ru
    Иначе
        Колонка = ЗагрузкаДанныхИзФайлаКлиентСервер.ОписаниеКолонкиМакета("ШтрихкодАртикул", ОбщегоНазначения.ОписаниеТипаСтрока(20), НСтр("ru
    КонецЕсли;
    Колонка.ОбязательнаДляЗаполнения = Истина;
    Колонка.Позиция = 1;
    Колонка.Группа = "Номенклатура";
    Колонка.Родитель = "Аналог";
    КолонкиМакета.Добавить(Колонка);

    Колонка = ЗагрузкаДанныхИзФайлаКлиентСервер.ОписаниеКолонкиМакета("Наименование", ОбщегоНазначения.ОписаниеТипаСтрока(100));
    Колонка.Группа = "Номенклатура";
    Колонка.Родитель = "Аналог";
    Колонка.Позиция = 2;
    Колонка.Подсказка = НСтр("ru='Наименование аналогичного товара, который полностью идентичен
    |по своему функциональному назначению и техническим характеристикам.'");
    КолонкиМакета.Добавить(Колонка);

    Колонка = ЗагрузкаДанныхИзФайлаКлиентСервер.КолонкаМакета("Совместимость", КолонкиМакета);
    Колонка.Позиция = 3;

    Возврат КолонкиМакета;
КонецФункции
```

См. полностью пример реализации в справочнике `_ДемоНоменклатура` в демонстрационной конфигурации.

Загрузка данных в объект с реквизитом «Идентификатор»

Если объект содержит реквизит с зарезервированным именем `Идентификатор`, то стандартная загрузка данных в этот реквизит не предусмотрена. Для загрузки данных в этот реквизит необходимо:

- определить табличный макет для загрузки данных согласно инструкциям выше;
- в табличном макете создать колонку с другим именем, но описывающим поведение колонки `Идентификатор`, например `ИдентификаторНоменклатуры`;
- при сопоставлении данных и в момент помещения данных в объект в процедурах `СопоставитьЗагружаемыеДанныеИзФайла` и `ЗагрузитьИзФайла` использовать загруженные данные колонки `ИдентификаторНоменклатуры` таблицы `ЗагружаемыеДанные`.

Это ограничение обусловлено тем, что механизм загрузки использует одноименную колонку для служебных целей.

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Загрузка данных из файлов в справочники.

Для загрузки данных в табличные части документов не требуются какие-либо дополнительные роли. Команды загрузки доступны тем пользователям, у которых есть права на редактирование самого документа.

Использование при разработке конфигурации

Разработка дополнительных обработок для загрузки данных

Для расширения возможностей загрузки данных из файла в справочники можно также использовать средства подсистемы **Дополнительные отчеты и обработки**.

Надо создать внешнюю обработку согласно требованиям раздела Дополнительные отчеты и обработки. Затем в функции `СведенияОВнешнейОбработке` модуля объекта обработки описать параметры команды. Например, для добавления варианта загрузки **Демо: Контрагенты (Юридические лица с контактной информацией)**:

```
// Возвращает сведения о внешней обработке.
Функция СведенияОВнешнейОбработке() Экспорт

    ПараметрыРегистрации = ДополнительныеОтчетыИОбработки.СведенияОВнешнейОбработке("2.2.3.1");

    ПараметрыРегистрации.Вид = ДополнительныеОтчетыИОбработкиКлиентСервер.ВидОбработкиДополнительнаяОбработка();
    ПараметрыРегистрации.Версия = "1.0";

    НоваяКоманда = ПараметрыРегистрации.Команды.Добавить();
    НоваяКоманда.Представление = НСтр("ru = 'Демо: Контрагенты (Юридические лица с контактной информацией)'");
    НоваяКоманда.Использование = ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыЗагрузкаДанныхИзФайла();
    НоваяКоманда.Модификатор   = Метаданные.Справочники._ДемоКонтрагенты.ПолноеИмя();
    НоваяКоманда.Идентификатор = "ЮрЛицаКонтрагентов";

    Возврат ПараметрыРегистрации;
КонецФункции
```

В модуль объекта обработки разместить экспортные процедуры `ОпределитьПараметрыЗагрузкиДанныхИзФайла`, где переопределяются настройки загрузки по умолчанию, и `СопоставитьЗагружаемыеДанныеИзФайла` и `ЗагрузитьИзФайла`, в которых реализовать логику сопоставления и загрузки данных в реквизиты справочника:

```
// Определяет параметры загрузки данных из файла.
//
// Параметры:
//   ИдентификаторКоманды – Строка – Имя команды, определенное в функции СведенияОВнешнейОбработке().
//   ПараметрыЗагрузки – Структура – Настройки загрузки данных:
//       * ИмяМакетаСШаблоном – Строка – Имя макета с шаблоном загружаемых данных.
//           По умолчанию используется макет "ЗагрузкаИзФайла".
//       * ОбязательныеКолонкиМакета - Массив - Список имен колонок обязательных для заполнения.
//
Процедура ОпределитьПараметрыЗагрузкиДанныхИзФайла(ИдентификаторКоманды, ПараметрыЗагрузки) Экспорт
    Если ИдентификаторКоманды = "ЮрЛицаКонтрагентов" Тогда
        ПараметрыЗагрузки.ИмяМакетаСШаблоном = "ЗагрузкаИзФайлКонтрагенты";
    КонецЕсли;
КонецПроцедуры
// Производит сопоставление данных, загружаемых из файла, с данными в информационной базе
//
//   ИдентификаторКоманды – Строка – Идентификатор команды
//   ЗагружаемыеДанные – ТаблицаЗначений – таблица значений с загружаемыми данными:
//     * СопоставленныйОбъект – СправочникСсылка.Ссылка – Ссылка на сопоставленный объект. Заполняется внутри процедуры
//     * <другие колонки>    – Строка – Состав колонок соответствует макету "ЗагрузкаИзФайла"
//
Процедура СопоставитьЗагружаемыеДанныеИзФайла(ИдентификаторКоманды, ЗагружаемыеДанные) Экспорт
КонецПроцедуры
// Производит загрузку строки сопоставленных данных в элемент справочника.
//
//   ИдентификаторКоманды – Строка – Идентификатор команды
//   ЗагружаемыеДанные – Структура – Ключ – название колонки, Значение – Данные ячейки
//     * СопоставленныйОбъект – СправочникСсылка.Ссылка – Ссылка на сопоставленный объект
//     * <другие колонки>    – Строка – Строки загружаемого файла
//   ПараметрыЗагрузки – Структура – Параметры с пользовательскими установками загрузки данных
//       * Обновлять – Булево – Требуется ли создавать новые объекты
//       * Создавать – Булево – Требуется ли обновлять новые объекты
// Отказ                     – Булево       – Отмена загрузки
//
Процедура ЗагрузитьИзФайла(ИдентификаторКоманды, ЗагружаемыеДанные, ПараметрыЗагрузки, Отказ) Экспорт
КонецПроцедуры
```

См. пример обработки загрузки данных в демонстрационной конфигурации: раздел **Администрирование** – **Печатные формы, отчеты и обработки** – **Дополнительные отчеты и обработки** – **Загрузка юридических лиц (с контактной информацией)**.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Заметки пользователя

Подсистема **Заметки пользователя** предназначена для хранения персональных заметок (различной неструктурированной информации, которая недоступна для других пользователей информационной базы). Заметки можно отмечать цветом, помещать на рабочий стол и объединять в группы.

Настройка

Для использования подсистемы необходимо:

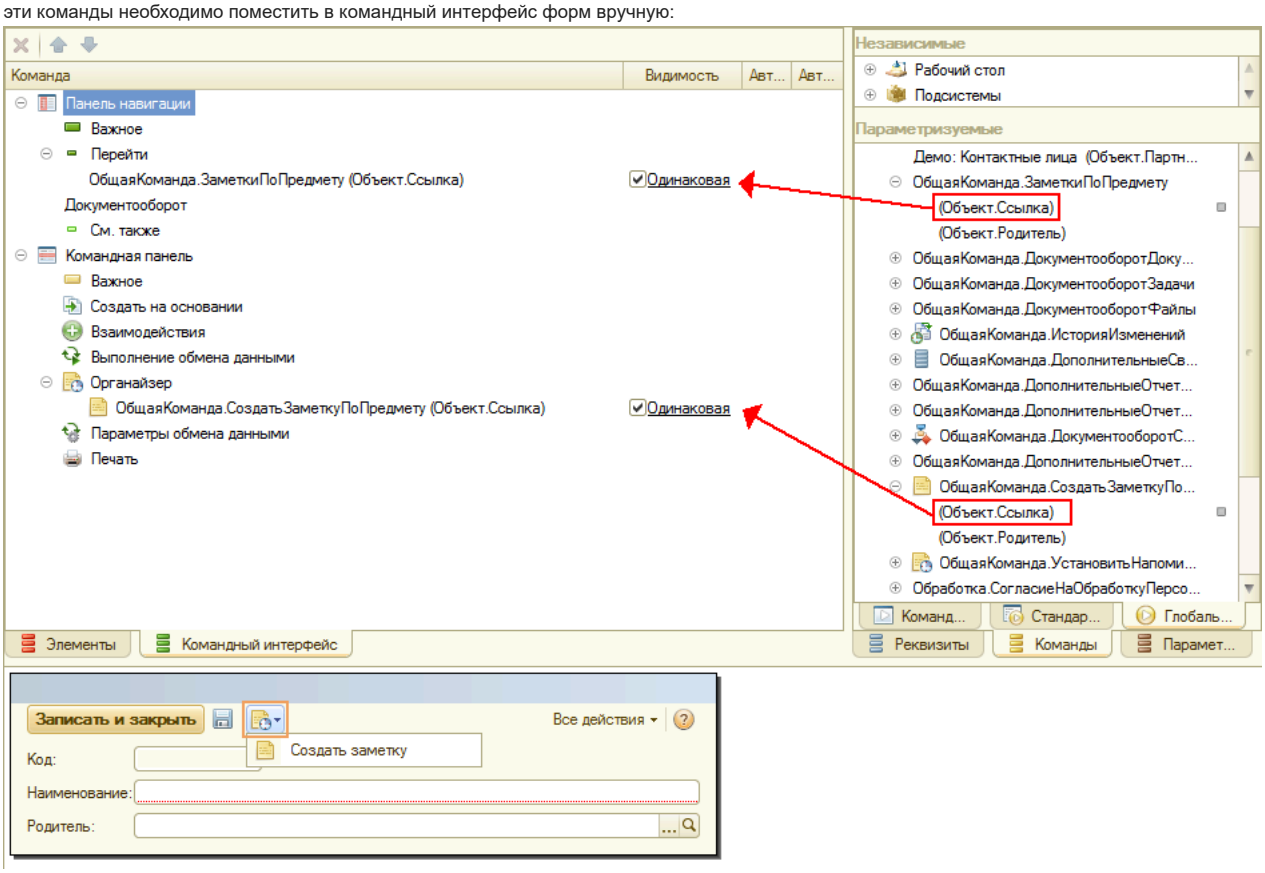
1. Определить состав объектов конфигурации, по поводу которых пользователи должны иметь возможность вводить заметки. Как правило, это большинство справочников.
2. Отметить выбранные типы объектов в свойстве `Тип` определяемых типов `ПредметЗаметок` (ссылки) и `ПредметЗаметокОбъект` (объекты).
3. Создать две подписки на событие `ПередЗаписью`:

- `УстановитьСтатусИзмененияПометкиУдаленияОбъекта`, в источнике указать выбранные типы объектов (кроме документов), обработчик – процедуру `ЗаметкиПользователя.УстановитьСтатусИзмененияПометкиУдаленияОбъекта`.
- `УстановитьСтатусИзмененияПометкиУдаленияДокумента`, в источнике указать выбранные типы документов, обработчик – процедуру `ЗаметкиПользователя.УстановитьСтатусИзмененияПометкиУдаленияДокумента`.

1. Разместить форму `МоиЗаметки` справочника `Заметки` на начальной странице, команду `ВсеЗаметки` справочника `Заметки` в командном интерфейсе пользователя.
2. Если в конфигурацию включена подсистема `ПодключаемыеКоманды`, то команды заметок выводятся с ее помощью. Для этого в тех формах, в которых должны выводиться команды заметок, дополнительно следует внедрить подсистему `ПодключаемыеКоманды`.

Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд напоминаний по шаблону:

- Имя: `ПодменюОрганайзер`.
- Заголовок: **Органайзер**.
- Вид: `Подменю`.
- Отображение: `Картинка`.
- Картинка: `Органайзер` (картинка из конфигурации).

> **Примечание**
>
> В случае отсутствия подсистемы `ПодключаемыеКоманды` в конфигурации, команды заметок выводятся с помощью общих команд. Общие команды `СоздатьЗаметкуПоПредмету` и `МоиЗаметкиПоПредмету` не появятся в формах групп объектов автоматически. При такой потребности эти команды необходимо поместить в командный интерфейс форм вручную:
>
> 
>

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Заметки пользователя» без подсистемы «Управление доступом».

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Создание, редактирование своих заметок, просмотр заметок пользователей, удаление заметок.
2.	`ДобавлениеИзменениеЗаметок` Создание, просмотр и пометка на удаление своих заметок.

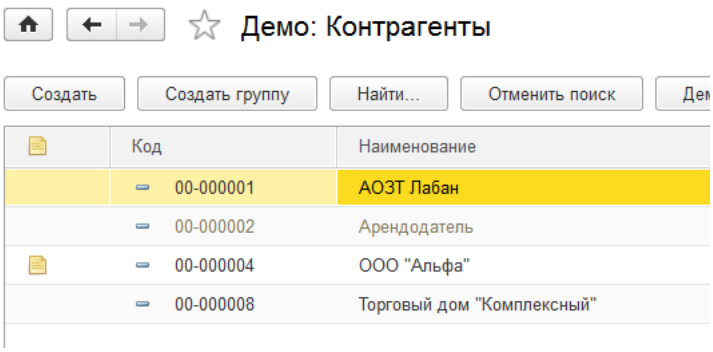
Использование при разработке конфигурации

Для отображения информации о наличии заметок непосредственно в форме списка объекта рекомендуется предусмотреть в конфигурации вспомогательный регистр сведений, в котором программно поддерживается информация о наличии заметок по объекту. У всех пользователей без ограничений должен быть предусмотрен доступ к этому регистру на просмотр. По сравнению с конструкцией `Когда…Тогда` с непосредственной выборкой из справочника **Заметки** при таком подходе к запросу не будут добавлены «тяжеловесные» ограничения доступа к данным на уровне записей (RLS), что позволяет повысить скорость открытия списка.

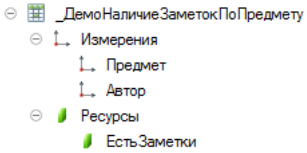
Для реализации такого подхода реквизит формы с динамическим списком необходимо переключить в режим **Произвольный запрос** и изменить текст запроса, добавив поле, как в примере в форме списка справочника `_ДемоКонтрагенты`:

```
ВЫБРАТЬ
    Справочник_ДемоКонтрагенты.Ссылка,
    Справочник_ДемоКонтрагенты.ПометкаУдаления,
    Справочник_ДемоКонтрагенты.Предопределенный,
    Справочник_ДемоКонтрагенты.Код,
    Справочник_ДемоКонтрагенты.Наименование,
    ВЫБОР
        КОГДА ЕСТЬNULL(_ДемоНаличиеЗаметокПоПредмету.ЕстьЗаметки, ЛОЖЬ)
            ТОГДА 0
        ИНАЧЕ -1
    КОНЕЦ КАК ЕстьЗаметки
ИЗ
    Справочник._ДемоКонтрагенты КАК Справочник_ДемоКонтрагенты
        ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений._ДемоНаличиеЗаметокПоПредмету КАК _ДемоНаличиеЗаметокПоПредмету
        ПО (Справочник_ДемоКонтрагенты.Ссылка = _ДемоНаличиеЗаметокПоПредмету.Предмет
                И _ДемоНаличиеЗаметокПоПредмету.Автор = &Пользователь)
```

Пример списка с колонкой-индикатором наличия заметок:



Пример структуры вспомогательного регистра сведений:



Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Запрет редактирования реквизитов объектов

Подсистема **Запрет редактирования реквизитов объектов** предназначена для запрета редактирования некоторых реквизитов объектов, которые определяют характер данного объекта и которые условно можно назвать ключевыми реквизитами данного объекта. Как правило, ключевыми реквизитами являются реквизиты объекта, влияющие на поведение других объектов, например, на проведение документов. Необдуманное изменение таких реквизитов может привести к рассогласованию данных учета.

Примером может служить реквизит `Валюта` справочника **Кассы**, который определяет валюту остатков по данной кассе, или признак **Является услугой** справочника **Номенклатура**, который влияет на проведение документов с такой номенклатурой. Такие реквизиты нельзя изменять после появления остатков по данной кассе или после появления документов с данной номенклатурой соответственно.

При этом подсистема допускает программное изменение ключевых реквизитов объектов. В этом случае необходимые проверки должен выполнять вызывающий код.

Настройка

Для использования подсистемы необходимо выполнить следующие шаги:

1. Определить список объектов, для которых применяется подсистема, и состав их ключевых реквизитов (как минимум один реквизит для каждого объекта). Перечислить список этих объектов в процедуре `ПриОпределенииОбъектовСЗаблокированнымиРеквизитами` общего модуля `ЗапретРедактированияРеквизитовОбъектовПереопределяемый`.

```
// Определить объекты метаданных, в модулях менеджеров которых ограничивается возможность
// редактирования реквизитов с помощью экспортной функции ПолучитьБлокируемыеРеквизитыОбъекта.
//
// Функция ПолучитьБлокируемыеРеквизитыОбъекта должна возвращать значение типа
// см. ЗапретРедактированияРеквизитовОбъектов.ОписаниеБлокируемогоРеквизита
//
// Поле надписи, связанное с реквизитом, не блокируется. Если требуется блокировать,
// имя элемента надписи нужно указать в описании блокируемого реквизита.
//
// Параметры:
//   Объекты - Соответствие из КлючИЗначение:
//     * Ключ - Строка - полное имя объекта метаданных, подключенного к подсистеме;
//     * Значение - Строка - пустая строка.
//
// Пример:
//   Объекты.Вставить(Метаданные.Документы.ЗаказПокупателя.ПолноеИмя(), "");
//
//   При этом в модуле менеджера объекта размещается код, подобный:
//   // См. ЗапретРедактированияРеквизитовОбъектовПереопределяемый.ПриОпределенииЗаблокированныхРеквизитов.ЗаблокированныеРеквизиты
//   Функция ПолучитьБлокируемыеРеквизитыОбъекта() Экспорт
//      БлокируемыеРеквизиты = Новый Массив;
//      БлокируемыеРеквизиты.Добавить("Организация");
//      БлокируемыеРеквизиты.Добавить("Партнер;Партнер");
//      Реквизит = ЗапретРедактированияРеквизитовОбъектов.НовыйБлокируемыйРеквизит();
//      Реквизит.Имя = "Контрагент";
//      Реквизит.Предупреждение = НСтр("ru = 'Поле не рекомендуется менять, если уже есть введенные документы'");
//      БлокируемыеРеквизиты.Добавить(Реквизит);
//      ...
//      Возврат БлокируемыеРеквизиты;
//   КонецФункции
//
Процедура ПриОпределенииОбъектовСЗаблокированнымиРеквизитами(Объекты) Экспорт
```

2. Принять решение по поводу состава объектов метаданных, наличие ссылок из которых на данный объект метаданных не должно влиять на возможность редактирования его ключевых реквизитов. Такие объекты следует включить в фильтр поиска исключений ссылок (см. раздел Настройка исключений поиска ссылок на объекты).

Для каждого объекта метаданных, для которого применяется подсистема, надо выполнить следующие шаги:

1. Добавить в модуль менеджера объекта функцию `ПолучитьБлокируемыеРеквизитыОбъекта`, в которой вернуть список имен ключевых реквизитов/табличных частей объекта или реквизитов формы. После имени реквизита можно указать точку с запятой и через запятую перечислить имена дополнительных элементов формы, имеющих отношение к реквизиту, которые тоже нужно заблокировать:

```
// См. ЗапретРедактированияРеквизитовОбъектовПереопределяемый.ПриОпределенииОбъектовСЗаблокированнымиРеквизитами.
Функция ПолучитьБлокируемыеРеквизитыОбъекта() Экспорт
    Результат = Новый Массив;
    Результат.Добавить("ИмяРеквизитаОбъекта1");
    Результат.Добавить("ИмяТабличнойЧастиОбъекта1");
    Результат.Добавить("ИмяРеквизитаФормы1");
    Результат.Добавить("ИмяРеквизитаОбъекта2; ИмяЭлементаФормы1, ИмяЭлементаФормы2");
    …
    Возврат Результат;
КонецФункции
```

   Поле надписи, связанное с реквизитом, не блокируется. Если требуется блокировать, имя элемента поля надписи нужно указать после точки с запятой, как написано выше. Можно также управлять внешним видом общей формы `РазблокированиеРеквизитов`: указать текст верхней надписи в шапке формы, тексты предупреждений для каждого реквизита, а также выводить реквизиты в группах. Пример:

```
// См. ЗапретРедактированияРеквизитовОбъектовПереопределяемый.ПриОпределенииОбъектовСЗаблокированнымиРеквизитами.
Функция ПолучитьБлокируемыеРеквизитыОбъекта() Экспорт
    БлокируемыеРеквизиты = Новый Массив;

    Реквизит = ЗапретРедактированияРеквизитовОбъектов.НовыйБлокируемыйРеквизит();
    Реквизит.Группа = "ОбщаяНадпись";
    Реквизит.ПредставлениеГруппы = НСтр("ru = 'Проверьте места использования перед разблокировкой реквизитов.'");
    БлокируемыеРеквизиты.Добавить(Реквизит);

    Реквизит = ЗапретРедактированияРеквизитовОбъектов.НовыйБлокируемыйРеквизит();
    Реквизит.Имя = "Код";
    Реквизит.Предупреждение = НСтр("ru = 'Это предупреждение для поля Код'");
    БлокируемыеРеквизиты.Добавить(Реквизит);

    Реквизит = ЗапретРедактированияРеквизитовОбъектов.НовыйБлокируемыйРеквизит();
    Реквизит.Группа = "НастройкиСчета";
    Реквизит.ПредставлениеГруппы = НСтр("ru = 'Настройки счета'");
    Реквизит.Предупреждение = НСтр("ru = 'Настройки счета не рекомендуется изменять, если счет уже используются.'");
    БлокируемыеРеквизиты.Добавить(Реквизит);

    БлокируемыеРеквизиты.Добавить("Вид;;НастройкиСчета");
    БлокируемыеРеквизиты.Добавить("Забалансовый;;НастройкиСчета");
    …
    Возврат Результат;
КонецФункции
```

2. Добавить в модуль формы объекта процедуру:

```
&НаКлиенте
Процедура Подключаемый_РазрешитьРедактированиеРеквизитовОбъекта(Команда)
    ЗапретРедактированияРеквизитовОбъектовКлиент.РазрешитьРедактированиеРеквизитовОбъекта(ЭтотОбъект);
КонецПроцедуры
```

3. Добавить в обработчики событий `ПриСозданииНаСервере` и `ПослеЗаписиНаСервере` код:

```
// Обработчик подсистемы запрета редактирования реквизитов объектов
ЗапретРедактированияРеквизитовОбъектов.ЗаблокироватьРеквизиты(ЭтотОбъект);
```

> **Примечание**
>
> Вторым параметром функции может выступать группа формы, на которую будет прикреплена команда разрешения редактирования ключевых реквизитов.

4. Для ключевых реквизитов в свойствах объекта рекомендуется выставлять свойство **Проверка заполнения** как **Выдавать ошибку** либо вручную осуществлять проверку ключевых реквизитов на заполненность.

Пример использования подсистемы можно посмотреть в демонстрационной конфигурации в справочнике `_ДемоНоменклатура`, плане счетов `_ДемоОсновной` и документе `_ДемоЗаказПокупателя`.

Переопределение логики, разрешающей изменение ключевых реквизитов

Существует возможность задать нестандартные условия разрешения изменения ключевых реквизитов. Для этого нужно создать в интересующем объекте форму `РазблокированиеРеквизитов`. В ней позволить пользователю пометить нужные реквизиты и вернуть результат с помощью метода `Закрыть` (`Массив` имен реквизитов для разблокирования). Пример:

```
&НаКлиенте
Процедура РазрешитьРедактирование(Команда)

    РазблокируемыеРеквизиты = Новый Массив;
    РазблокируемыеРеквизиты.Добавить("<Имя реквизита>");
    …
    Закрыть(РазблокируемыеРеквизиты);

КонецПроцедуры
```

При совместном внедрении с подсистемой **Групповое изменение объектов** рекомендуется также реализовать в форме `РазблокированиеРеквизитов` вариант вызова для разблокировки реквизитов при групповом изменении объектов. Для этого в ней следует определить три параметра:

- `ГрупповоеИзменениеОбъектов` типа `Булево`,
- `ЗаблокированныеРеквизиты` типа `Произвольный`,
- `АдресСсылокНаОбъекты` типа `Строка`.

При этом в форму передаются параметры `ГрупповоеИзменениеОбъектов` со значением `Истина`, `ЗаблокированныеРеквизиты` с массивом имен реквизитов для разблокирования и `АдресСсылокНаОбъекты` с адресом временного хранилища с массивом ссылок для проверки использования объектов (может быть не заполнен).

Если в форме `РазблокированиеРеквизитов` не определен параметр `ГрупповоеИзменениеОбъектов`, то при групповом изменении объектов вместо нее будет открываться стандартная общая форма `РазблокированиеРеквизитов`.

В редких случаях также бывает необходимо самостоятельно рассчитывать и устанавливать значения свойств `ТолькоПросмотр` или `Доступность` соответствующих полей разблокируемых реквизитов. В таких случаях процедура `Подключаемый_РазрешитьРедактированиеРеквизитовОбъекта` модуля формы объекта реализуется иначе. Пример:

```
&НаКлиенте
Процедура Подключаемый_РазрешитьРедактированиеРеквизитовОбъекта(Команда)
    ЗаблокированныеРеквизиты = ЗапретРедактированияРеквизитовОбъектовКлиент.Реквизиты(ЭтотОбъект);

    Если ЗаблокированныеРеквизиты.Количество() > 0 Тогда
        ПараметрыФормы = Новый Структура;
        ПараметрыФормы.Вставить("Ссылка", Объект.Ссылка);
        ПараметрыФормы.Вставить("ЗаблокированныеРеквизиты", ЗаблокированныеРеквизиты);

        ОткрытьФорму("Документ._ДемоЗаказПокупателя.Форма.РазблокированиеРеквизитов", ПараметрыФормы,
            ЭтотОбъект,,,, Новый ОписаниеОповещения("ПослеВыбораРеквизитовДляРазблокирования", ЭтотОбъект));
    Иначе
        ЗапретРедактированияРеквизитовОбъектовКлиент.ПоказатьПредупреждениеВсеВидимыеРеквизитыРазблокированы();
    КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ПослеВыбораРеквизитовДляРазблокирования(РазблокируемыеРеквизиты, Контекст) Экспорт

    Если ТипЗнч(РазблокируемыеРеквизиты) <> Тип("Массив") Тогда
        Возврат;
    КонецЕсли;

    ЗапретРедактированияРеквизитовОбъектовКлиент.УстановитьРазрешенностьРедактированияРеквизитов(ЭтотОбъект,
        РазблокируемыеРеквизиты);

    ... // Код по отключению свойства ТолькоПросмотр и включению свойства Доступность у элементов.

КонецПроцедуры
```

При этом в обработчике `ПослеВыбораРеквизитовДляРазблокирования` размещается вызов процедуры `ЗапретРедактированияРеквизитовОбъектовКлиент.УстановитьРазрешенностьРедактированияРеквизитов`, которая только отмечает реквизиты как разрешенные для редактирования, но не изменяет доступность полей реквизитов. Кроме того, если какие-то из реквизитов нужно считать недоступными для разблокировки (например, если для реквизита формы нет права редактирования, заданного через роли, или по другой причине), то можно переопределить значение свойства `ПравоРедактирования` с помощью процедуры `ЗапретРедактированияРеквизитовОбъектовКлиент.УстановитьРазрешенностьРедактированияРеквизитов`.

Размещение в командном интерфейсе

Все действия по изменению командного интерфейса выполняются при интеграции подсистемы (см. выше).

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Запрет редактирования реквизитов объектов** следует использовать роль, указанную ниже.

№	Роли и их назначение
1.	`РедактированиеРеквизитовОбъектов`. Редактирование заблокированных значений реквизитов объектов.

Пример настройки прав доступа пользователей приведен ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за ведение документов	• `ДобавлениеИзменениеДокументов` – основная роль, предоставляющая право редактирования документа. • `РедактированиеРеквизитовОбъектов`.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Защита персональных данных

Подсистема **Защита персональных данных** предназначена для обеспечения соответствия информационной системы, построенной на основе конфигурации, требованиям Федерального закона № 152-ФЗ от 27.06.2006 «О персональных данных».

Настройка

Для внедрения подсистемы **Защита персональных данных** необходимо принять решение по поводу того, какие сущности в конфигурации являются субъектами персональных данных, т. е. к каким сущностям относятся защищаемые персональные данные. Как правило, это такие справочники, как **Физические лица**, **Сотрудники**, **Контрагенты**, **Партнеры** и др.

Затем необходимо задать состав типов субъектов персональных данных в качестве определяемого типа `СубъектПерсональныхДанных`.

В общем модуле `ЗащитаПерсональныхДанныхПереопределяемый` в процедуре `ДополнитьДанныеСубъектовПерсональныхДанных` описать определение данных субъекта, необходимых для заполнения документа согласия на обработку персональных данных. См. пример реализации этой процедуры в демонстрационной конфигурации.

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить обработку `ЗащитаПерсональныхДанных` и ее команду `ЗащитаПерсональныхДанных`. См. пример в форме `ПоддержкаИОбслуживание` обработки `ПанельАдминистрированияБСП`.

Также необходимо принять решение о размещении в интерфейсе рабочего места для уничтожения персональных данных (обработки `УничтожениеПерсональныхДанных`). См. пример в демонстрационной конфигурации в разделе **Интегрируемые подсистемы**.

Настройка регистрации доступа к персональным данным

После этого необходимо включить регистрацию событий журнала регистрации типа `Доступ.Доступ`. Для этого в процедурах `ЗаполнитьСведенияОПерсональныхДанных` и `ЗаполнитьКатегорииПерсональныхДанных` общего модуля `ЗащитаПерсональныхДанныхПереопределяемый` реализовать заполнение сведений о персональных данных и категориях персональных данных.

Любая информация, содержащая ФИО и иные атрибуты физического лица (ИНН, номер паспорта, водительского или иного удостоверения, телефон и т.д.) относится к персональным данным.

Описание сведений о персональных данных представляет собой таблицу значений с колонками типа `Строка`:

- `Объект` – полное имя объекта метаданных;

- `ПоляРегистрации` – имена полей регистрации, разделенные запятой. При необходимости задать альтернативные поля их можно перечислить, разделяя символом `|`;

- `ПоляДоступа` – имена полей доступа, разделенные запятой;
- `ОбластьДанных` – имя категории персональных данных, к которой относятся описываемые сведения. Например, `"ФИО"`, `"ДатаМестоРождения"`, `"ПаспортныеДанные"`.

Описание категорий также заполняется в виде таблицы значений с колонками типа `Строка`:

- `Имя` – имя категории данных (так, как указано в таблице сведений);
- `Представление` – пользовательское представление категории;
- `Родитель` – данные, в состав которых входит описываемая категория.

Логика заполнения сведений о персональных данных и категорий должна определяться исходя из состава субъектов персональных данных, а также состава данных информационной базы, относящихся к персональным. Пример такого заполнения см. в процедурах `ЗаполнитьСведенияОПерсональныхДанных` и `ЗаполнитьКатегорииПерсональныхДанных` общего модуля `ЗащитаПерсональныхДанныхПереопределяемый`.

События `Доступ.Доступ` и `Доступ.Отказ в доступе` выводятся в форме обработки `ЗащитаПерсональныхДанных` в том случае, если в информационной базе включена их регистрация. При этом следует иметь в виду, что событие `Доступ.Отказ в доступе` по умолчанию включено.

Для управления регистрацией событий доступа к персональным данным в форме настройки системы необходимо разместить реквизит и элементы для управления:

- добавить реквизит формы `КатегорииПерсональныхДанных` типа `ДеревоЗначений`,
- добавить таблицу формы `КатегорииПерсональныхДанных`, связанную с этим реквизитом.

Разместить вызовы процедур общего модуля `ЗащитаПерсональныхДанных`:

- в обработчике `ПриСозданииНаСервере` – `ПриСозданииФормыНастройкиРегистрацииСобытий`,
- в обработчике `ПриЗаписиНаСервере` – `ПриЗаписиФормыНастройкиРегистрацииСобытий`.

Далее необходимо определить место в командном интерфейсе для включения объектов работы с согласиями на обработку персональных данных субъектов:

- журнала `СогласияНаОбработкуПерсональныхДанных`,
- отчетов `СогласияНаОбработкуПерсональныхДанныхДействующие` и `СогласияНаОбработкуПерсональныхДанныхИстекающие`.

Настройка уничтожения персональных данных

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора приложения необходимо разместить константу `ИспользоватьСкрытиеПерсональныхДанныхСубъектов`.

См. пример размещения в демонстрационной конфигурации в группе **Защита персональных данных** формы `НастройкиПользователейИПрав` обработки `ПанельАдминистрированияБСП`.

Согласие на обработку персональных данных

Для отслеживания потребности в получении согласия на обработку персональных данных и отбора субъектов с уничтоженными персональными данными необходимо в формах списка субъектов (например, физических лиц), определенных в определяемом типе `СубъектПерсональныхДанных`, выполнить ряд действий:

1. В обработчике события формы `ПриСозданииНаСервере` добавить код:

```
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
ЗащитаПерсональныхДанных.ПриСозданииНаСервереФормыСписка(ЭтотОбъект, Элементы.Список);
// Конец СтандартныеПодсистемы.ЗащитаПерсональныхДанных
```

2. В обработчике события формы `ОбработкаОповещения` добавить:

```
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
ЗащитаПерсональныхДанныхКлиент.ОбработкаОповещенияФормыСписка(Элементы.Список, ИмяСобытия);
// Конец СтандартныеПодсистемы.ЗащитаПерсональныхДанных
```

3. В обработчике события таблицы формы `СписокПриПолученииДанныхНаСервере` добавить код:

```
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
ЗащитаПерсональныхДанных.ПриПолученииДанныхНаСервере(Настройки, Строки);
// Конец СтандартныеПодсистемы.ЗащитаПерсональныхДанных
```

4. Дополнительно для управления видимостью субъектов с ранее скрытыми персональными данными добавляется подключаемая процедура:

```
#Область ОбработчикиКомандФормы
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
&НаКлиенте
Процедура Подключаемый_ПоказыватьСоСкрытымиПДн(Команда) Экспорт
    ЗащитаПерсональныхДанныхКлиент.ПоказыватьСоСкрытымиПДн(ЭтотОбъект, Список);
КонецПроцедуры
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
#КонецОбласти
```

5. Помимо встраивания кода необходимо настроить элементы и свойства формы списка. Расположить на командной панели группу с именем `ГруппаПоказыватьСоСкрытымиПДн`.

6. Настроить динамический список формы, добавив соответствующее поле и псевдоним:

```
ВЫРАЗИТЬ(NULL КАК ЧИСЛО(1, 0)) КАК ОтсутствуетСогласие,
```

7. К запросу динамического списка добавить соединение с таблицей регистра сведений `СубъектыДляСкрытияПерсональныхДанных`:

```
ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.СубъектыДляСкрытияПерсональныхДанных КАК СубъектыСоКрытымиПДн
        ПО Справочник_ДемоФизическиеЛица.Ссылка = СубъектыСоКрытымиПДн.Субъект
ГДЕ
    (СубъектыСоКрытымиПДн.Субъект ЕСТЬ NULL
            ИЛИ &ПоказыватьСоСкрытымиПДн)
```

   Готовый пример реализации см. в форме списка справочника `_ДемоФизическиеЛица` демонстрационной конфигурации.

8. По желанию для управления местом размещения колонки **Отсутствует требуемое согласие** можно добавить элемент поля таблицы `ОтсутствуетСогласие`.

Описание уничтожаемых персональных данных

Состав объектов информационной базы и их полей, в которых содержатся персональных данные субъектов, описывается в процедуре `ПриЗаполненииСведенийОбУничтожаемыхПерсональныхДанных` общего модуля `ЗащитаПерсональныхДанныхПереопределяемый`.

Описание представляет собой таблицу значений с колонками:

- `Объект` – тип `Строка` - полное имя объекта метаданных, в котором содержатся персональные данные субъекта;
- `ПолеСубъект` – тип `Строка` – имя поля объекта, в котором содержится ссылка на субъект. Для стандартных реквизитов используется имя реквизита, например `Ссылка`. Для остальных полей используется полное имя объекта метаданных реквизита, например `Документ.ВедомостьНаВыплатуЗарплатыВБанк.ТабличнаяЧасть.Состав.Реквизит.ФизическоеЛицо`;
- `Поля` – тип `Массив из Строка` - имена полей объекта, в которых содержатся персональные данные субъекта. Для стандартных реквизитов используется имя реквизита, например `Наименование`. Для остальных полей используется полное имя объекта метаданных реквизита, например `Справочник.ФизическиеЛица.Реквизит.ФИО`.
- `КатегорияДанных` – тип `Строка` – имя категории персональных данных, к которой относятся описываемые сведения.

Уничтожение персональных данных, хранимых в ссылочных метаданных (`Справочники`, `Документы`), будет выполнено при соблюдении условий:

- сведения о персональных данных объекта описаны в таблице уничтожаемых персональных данных;
- если субъект персональных данных встречается только в реквизитах объекта метаданных, то обрабатываются все строки табличных частей объекта;

- если ссылка на субъект есть хотя бы в одной табличной части, то для обработки остальных табличных частей она должна быть и там;
- если строки табличной части содержат данные разных субъектов, то для обработки с отбором по субъекту он должен быть в реквизитах этой табличной части.

Необходимо убедиться, что не требуется уточнения `ПолеСубъект` и `Поля` в существующих описаниях и пересмотра структуры объектов метаданных:

- убрать из `Поля` путь к `Общим` реквизитам объекта метаданных, которые не должны быть обработаны (возможно, удалить реквизиты из метаданных, заменив их в формах на динамически рассчитываемые);
- добавить в обрабатываемую табличную часть реквизит с типом субъекта (см. определяемый тип `СубъектПерсональныхДанных`), если такой отсутствует, но подразумевается. И указать его в `ПолеСубъект`.

Для обновления формы элемента субъекта персональных данных при наступлении события `УничтоженыПерсональныеДанные` в обработчике `ОбработкаОповещения` размещается вызов:

```
// СтандартныеПодсистемы.ЗащитаПерсональныхДанных
ЗащитаПерсональныхДанныхКлиент.ОбработкаОповещенияФормы(ЭтотОбъект, ИмяСобытия);
// Конец СтандартныеПодсистемы.ЗащитаПерсональныхДанных
```

При уничтожении персональных данных запись объекта выполняется в режиме `ОбменДанными.Загрузка`. Поэтому при необходимости дополнительных действий перед записью и после записи объекта можно использовать обработчики `ПередУничтожениемПерсональныхДанных` и `ПослеУничтоженияПерсональныхДанных` общего модуля `ЗащитаПерсональныхДанныхПереопределяемый`.

Сроки хранения персональных данных

Персональные данные субъекта могут быть уничтожены только по истечении их срока хранения. Расчет сроков хранения выполняется в процедуре `ПриРасчетеСроковХраненияПерсональныхДанных` общего модуля `ЗащитаПерсональныхДанныхПереопределяемый`. При этом заполняется таблица значений с колонками:

- `Субъект` - ссылка на субъекта.
- `СрокХранения` - дата, до которой должны храниться персональные данные субъекта и при наступлении которой, они должны быть уничтожены.
- `Организация` - ссылка на организацию субъекта (если организацию определить невозможно, не заполняется).
- `Комментарий` - произвольная строка с пояснением к рассчитанному сроку хранения.

Датой, при наступлении которой данные подлежат уничтожению, будет определена самая поздняя из всех заполненных для этого субъекта.

Сроки хранения персональных данных должны поддерживаться в актуальном состоянии. Поэтому при наступлении события, которое влияет на срок хранения, необходимо с помощью процедуры `ДобавитьСубъектыДляРасчетаСроковХранения` общего модуля `ЗащитаПерсональныхДанных` зарегистрировать субъект для расчета срока хранения. Такими событиями могут быть, например, создание нового физического лица, ввод согласия на обработку персональных данных или прием сотрудника на работу. После этого срок хранения будет рассчитан автоматически регламентным заданием `РасчетСроковХраненияПерсональныхДанных`.

При использовании механизма распределенных информационных баз (РИБ) расчет сроков хранения и уничтожение персональных данных субъектов возможны только в базе главного узла.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к инструментам подсистемы **Защита персональных данных** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ЧтениеАктовОбУничтожении ПерсональныхДанных` Просмотр актов об уничтожении персональных данных.
2.	`ДобавлениеИзменениеАктовОбУничтожении ПерсональныхДанных` Ввод и редактирование актов об уничтожении персональных данных, а также запуск уничтожения персональных данных.
3.	`ПросмотрЖурналаРегистрации` (из подсистемы **Базовая функциональность**) Просмотр событий доступа к персональными данными в журнале регистрации.
4.	`Администрирование` (из подсистемы **Базовая функциональность**)

Для доступа к документам **Согласие на обработку персональных данных**, **Отзыв согласия на обработку персональных данных** необходимо создать в конфигурации роли по аналогии с демонстрационными ролями `_ДемоДобавлениеИзменениеСогласийНаОбработкуПерсональныхДанных` и `_ДемоЧтениеСогласийНаОбработкуПерсональныхДанных`, описать в них текст необходимого ограничения доступа на уровне записей.

Если в форме списка субъектов требуется выделять субъектов, персональные данные которых уничтожены, следует описать доступ к регистру сведений `СубъектыДляСкрытияПерсональныхДанных` в одной из ролей, используемых для доступа к субъектам персональных данных.

Использование при разработке конфигурации

Настройка обмена данными

Все объекты метаданных подсистемы рекомендуется включать в планы обмена распределенной информационной базы (РИБ), автономных рабочих мест, а также в планы обмена, предназначенные для синхронизации данных между различными приложениями, за исключением следующих метаданных:

- константа `ИспользоватьСкрытиеПерсональныхДанныхСубъектов`,
- регистр сведений `ОбластиПерсональныхДанных`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, также исключить:

- регистр сведений `СогласияНаОбработкуПерсональныхДанных`.

В обменах через универсальный формат `EnterpriseData` при выгрузке данных, зарегистрированных на узле плана обмена, выполняется принудительная выгрузка объектов по ссылкам в том случае, если ранее они не выгружались. Если выгружаемые данные не относятся к источникам персональных данных, но в реквизитах присутствует ссылка на субъект или источник персональных данных, то необходимо позаботиться о том, чтобы такой объект не был выгружен, если он содержит в себе скрытые персональные данные. Это можно сделать на уровне правил обработки данных, воспользовавшись методом `ЭтоОбъектСУничтоженнымиПерсональнымиДанными` программного интерфейса модуля `ЗащитаПерсональныхДанных`.

Интерфейс OData

Стандартный интерфейс OData — это REST интерфейс прикладного решения, который можно опубликовать на веб-сервере, через который внешние приложения взаимодействуют с информационной базой с помощью HTTP-запросов. Он позволяет читать данные, изменять их, создавать новые объекты данных и удалять существующие. Подробнее см. документацию 1С:Предприятие 8.3. Руководство разработчика.

Подсистема **Интерфейс OData** дает дополнительные возможности:

- разработчику — ограничивать список объектов, доступ к которым может быть предоставлен таким образом с помощью роли `УдаленныйДоступOData`;
- администратору приложения — выбирать из этого списка, какие именно объекты будут доступны в приложении.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора необходимо разместить команду `НастройкиСтандартногоИнтерфейсаOData` обработки `НастройкаСтандартногоИнтерфейсаOData` (см. пример размещения в форме `Органайзер` обработки `ПанельАдминистрированияБСП` демонстрационной конфигурации).

В состав роли `УдаленныйДоступOData` необходимо включить те объекты конфигурации, чтение, изменение и удаление которых предполагается из внешних приложений. Рекомендуется включать все константы, справочники, документы и регистры конфигурации, а также связанные с ними перечисления и параметры сеанса. Исключение составляют:

- служебные и технологические метаданные, работа с которыми из внешних приложений не предполагается или недопустима;
- если конфигурация рассчитана на работу в модели сервиса, то в роль также не следует включать неразделенные объекты метаданных.

Для автоматической проверки и настройки роли `УдаленныйДоступOData` рекомендуется запускать отчет `ПроверкаВнедренияБСП`, который входит в состав дистрибутива библиотеки.

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Управлением доступом и настройка списка объектов, доступ к которым предоставлен внешним приложениям с помощью стандартного интерфейса **OData**.
2.	`УдаленныйДоступOData` Чтение, изменение и удаление данных прикладного решения из внешних приложений с помощью интерфейса **OData**.

Использование при разработке конфигурации

Настройка обмена данными

Константа `ПользовательСтандартногоИнтерфейсаOData` не должна включаться в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места. Значение этой константы задается независимо в каждом из узлов.

Информация при запуске

Подсистема **Информация при запуске** отображает HTML-страницы с различной информацией при запуске приложения (например, рекламу). Страницы содержатся в макетах обработки `ИнформацияПриЗапуске`. Каждый макет содержит стартовую страницу, а также может содержать другие страницы, ссылки которых указаны в стартовой странице. Для каждого макета на командной панели обработки автоматически создается команда, по щелчку по которой осуществляется переход к странице с информацией.

Настройка

Определить набор страниц с информацией, которая должна выводиться при запуске приложения.

Каждый набор страниц запаковать в ZIP-архив таким образом, чтобы стартовая страница была в корне этого архива.

Загрузить архивы в обработку `ИнформацияПриЗапуске`, добавив их в виде макетов из двоичных данных.

Описать настройки каждого макета (архива) в макете `Описатель` обработки `ИнформацияПриЗапуске`, указав свойства:

- **Имя макета** – имя макета, как оно задано в метаданных;
- **Раздел** – наименование подменю, в котором будет размещена команда вызова информации, размещенной в макете. Если наименование подменю не указано, то команда будет размещена прямо в командной панели;
- **Наименование** – наименование команды, по щелчку по которой будет осуществлен переход к странице с информацией;
- **Имя стартового файла** – имя стартовой HTML-страницы в ZIP-архиве. Указывается только имя, без слешей и имени архива;
- **Дата начала показа** – дата начала действия этой строки;
- **Дата окончания показа** – дата окончания действия этой строки;
- **Включать в показ при открытии** – используется для определения стартовой страницы, отображаемой при открытии обработки. Если «0» или «» (не указано), то не показывать при открытии. «1» – включать в первую очередь, «2» – включать во вторую очередь. При прочих равных (в случае возникновения конкуренции) страница выбирается случайным образом;
- **Показывать в ПРОФ** – если `Истина`, то информация будет показана в локальном режиме в версии ПРОФ;
- **Показывать в базовой** – если `Истина`, то информация будет показана в локальном режиме в базовой версии;
- **Показывать в модели сервиса** – если `Истина`, то информация будет показана в модели сервиса.

> **Внимание!**
>
> Для того чтобы изменения в макетах обработки `ИнформацияПриЗапуске` вступили в силу сразу при разработке и отладке конфигурации, необходимо обновить вспомогательные данные. Подробнее см. раздел Обновление вспомогательных данных во время разработки.

Настройка прав доступа пользователей

Для настройки прав доступа дополнительных действий не требуется.

№	Роли и их назначение
1.	`БазовыеПраваБСП или БазовыеПраваВнешнихПользователейБСП`. Просмотр информации.

Настройка обмена данными

Регистр сведений `ПакетыИнформацииПриЗапуске` рекомендуется включать только в состав начального образа подчиненного узла распределенной информационной базы (РИБ) и автономного рабочего места (см. раздел Особенности создания начального образа подчиненного узла распределенной ИБ).

Календарные графики

Подсистема **Календарные графики** предназначена для хранения графиков, по которым работает предприятие.

С подсистемой поставляются готовые функции, которые позволяют получать различную информацию по рабочим дням, входящим в график.

Настройка

Для использования подсистемы в конфигурации необходимо поместить справочник `ПроизводственныеКалендари` в командный интерфейс конфигурации.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Календарные графики** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Просмотр календарей и графиков, имеющихся в приложении.
2.	`ДобавлениеИзменениеКалендарныхГрафиков` Ведение календарей и графиков.
3.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Удаление помеченных на удаление объектов подсистемы.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Конструктор формул

Подсистема **Конструктор формул** предоставляет удобную форму редактирования формул, в которой выводятся доступные операнды и операторы. При наличии подсистемы **Мультиязычность** операнды и операторы формулы отображаются на текущем языке интерфейса.

Например, формула курса валюты `(USD + EUR) / 2`, хранящаяся в данных, в форме редактирования формулы может выглядеть как `([Доллар США] + [Евро]) / 2`.

Настройка

Настройка прав доступа пользователей

Для настройки прав доступа дополнительных действий не требуется.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**): Открытие формы редактирования формулы.

Использование при разработке конфигурации

Программный интерфейс подсистемы предусматривает два сценария:

1. открытие формы редактора,
2. встраивание списка доступных полей конструктора формул в другие формы.

Открытие поставляемой формы редактора

Для открытия формы редактирования формулы предусмотрен программный интерфейс в модулях `КонструкторФормулКлиент` и `КонструкторФормул`. Для открытия формы редактирования формулы, например, из поля ввода, следует вызвать процедуру `КонструкторФормулКлиент.НачатьРедактированиеФормулы`, и обработать результат выполнения после закрытия формы.

Пример:

```
&НаКлиенте
Процедура ФормулаРасчетаКурсаНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
  СтандартнаяОбработка = Ложь;
  ОписаниеОповещения = Новый ОписаниеОповещения("ПриЗавершенииРедактированияФормулы", ЭтотОбъект);
  КонструкторФормулКлиент.НачатьРедактированиеФормулы(ПараметрыФормулы(), ОписаниеОповещения);
КонецПроцедуры

&НаКлиенте
Процедура ПриЗавершенииРедактированияФормулы(ОписаниеФормулы, ДополнительныеПараметры) Экспорт
  Если ОписаниеФормулы = Неопределено Тогда
    Возврат;
  КонецЕсли;

  Объект.ФормулаРасчетаКурса = ОписаниеФормулы.Формула;
  ПредставлениеФормулы = ОписаниеФормулы.ПредставлениеФормулы;
  Модифицированность = Истина;
КонецПроцедуры
```

Операторы и операнды описываются единообразно в виде таблицы значений, дерева значений или схемы компоновки данных. При использовании таблицы значений и дерева значений возможно переопределение картинок реквизитов, а также порядка следования элементов.

```
&НаСервереБезКонтекста
Функция ТаблицаОперандов()

  ТаблицаОперандов = КонструкторФормул.ТаблицаПолей();

  Для Каждого ОписаниеВалюты Из Справочники.Валюты.КодыВалют() Цикл
    Операнд = ТаблицаОперандов.Добавить();
    Операнд.Идентификатор = ОписаниеВалюты.СимвольныйКод; // "USD"
    Операнд.Представление = ОписаниеВалюты.Представление; // "Доллар США"
    Операнд.ТипЗначения = Новый ОписаниеТипов("Число");
  КонецЦикла;

  Возврат ТаблицаОперандов;

КонецФункции
```

Для получения представления формулы, хранящейся в данных, например, для вывода на форме в виде гиперссылки, предусмотрена функция `КонструкторФормул.ПредставлениеФормулы`.

Встраивание списка доступных полей конструктора формул в другие формы

Форма редактора формулы представляет из себя поле для редактирования формулы и два списка - операторы и операнды. Списки обладают рядом полезных свойств: могут отображать неограниченную по уровням иерархию полей, автоматически подбирать картинки для полей, а также имеют поля поиска. Программный интерфейс подсистемы позволяет размещать такие списки в произвольной форме. Примером такого использования является форма выбора поля, открывающаяся из формы настроек отчета (см. форму `ВыборПоляОтчета` хранилища настроек `ХранилищеВариантовОтчетов`).

Для размещения подключаемого списка полей на форме необходимо выполнить вставку кода в модуль формы:

```
#Область ПодключаемыйСписокПолей

&НаКлиенте
Процедура Подключаемый_СписокПолейПередРазворачиванием(Элемент, Строка, Отказ)
  КонструкторФормулКлиент.СписокПолейПередРазворачиванием(ЭтотОбъект, Элемент, Строка, Отказ);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_РазвернутьТекущийЭлементСпискаПолей()
  КонструкторФормулКлиент.РазвернутьТекущийЭлементСпискаПолей(ЭтотОбъект);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ЗаполнитьСписокДоступныхПолей(ПараметрыЗаполнения) Экспорт
  ЗаполнитьСписокДоступныхПолей(ПараметрыЗаполнения);
КонецПроцедуры

&НаСервере
Процедура ЗаполнитьСписокДоступныхПолей(ПараметрыЗаполнения)
  КонструкторФормулСлужебный.ЗаполнитьСписокДоступныхПолей(ЭтотОбъект, ПараметрыЗаполнения);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_СписокПолейНачалоПеретаскивания(Элемент, ПараметрыПеретаскивания, Выполнение)
  КонструкторФормулКлиент.СписокПолейНачалоПеретаскивания(ЭтотОбъект, Элемент, ПараметрыПеретаскивания, Выполнение);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_СтрокаПоискаИзменениеТекстаРедактирования(Элемент, Текст, СтандартнаяОбработка)
  КонструкторФормулКлиент.СтрокаПоискаИзменениеТекстаРедактирования(ЭтотОбъект, Элемент, Текст, СтандартнаяОбработка);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ВыполнитьПоискВСпискеПолей()
    ВыполнитьПоискВСпискеПолей();
КонецПроцедуры

&НаСервере
Процедура ВыполнитьПоискВСпискеПолей()
    КонструкторФормул.ВыполнитьПоискВСпискеПолей(ЭтотОбъект);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_СтрокаПоискаОчистка(Элемент, СтандартнаяОбработка)
    КонструкторФормулКлиент.СтрокаПоискаОчистка(ЭтотОбъект, Элемент, СтандартнаяОбработка);
КонецПроцедуры

&НаСервере
Процедура Подключаемый_ОбработчикКонструктораФормулСервер(Параметр, ДополнительныеПараметры)
    КонструкторФормул.ОбработчикКонструктораФормул(ЭтотОбъект, Параметр, ДополнительныеПараметры);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ОбработчикКонструктораФормулКлиент(Параметр, ДополнительныеПараметры = Неопределено) Экспорт
    КонструкторФормулКлиент.ОбработчикКонструктораФормул(ЭтотОбъект, Параметр, ДополнительныеПараметры);
    Если ДополнительныеПараметры.ВыполнитьНаСервере Тогда
        Подключаемый_ОбработчикКонструктораФормулСервер(Параметр, ДополнительныеПараметры);
    КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_НачатьПоискВСпискеПолей()
    КонструкторФормулКлиент.НачатьПоискВСпискеПолей(ЭтотОбъект);
КонецПроцедуры

#КонецОбласти
```

Затем добавить вызов `КонструкторФормул.ДобавитьСписокПолейНаФорму` в обработчик события формы `ПриСозданииНаСервере`. Пример:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
  ПараметрыДобавленияСпискаПолей = КонструкторФормул.ПараметрыДобавленияСпискаПолей();
  ПараметрыДобавленияСпискаПолей.ИмяСписка = ИмяСпискаОперандов();
  ПараметрыДобавленияСпискаПолей.КоллекцииПолей.Добавить(ПрочитатьСписокОперандов());
  ПараметрыДобавленияСпискаПолей.МестоРазмещенияСписка = Элементы.ГруппаДоступныеПоля;
  ПараметрыДобавленияСпискаПолей.ПодсказкаВводаСтрокиПоиска = НСтр("ru = 'Найти операнд...'");
  ПараметрыДобавленияСпискаПолей.ИспользоватьИдентификаторыДляФормул = Истина;
  ПараметрыДобавленияСпискаПолей.ИспользоватьФоновыйПоиск = Истина;
  ПараметрыДобавленияСпискаПолей.ОбработчикиСписка.Вставить("Выбор", "Подключаемый_СписокПолейВыбор");
  КонструкторФормул.ДобавитьСписокПолейНаФорму(ЭтотОбъект, ПараметрыДобавленияСпискаПолей);
КонецПроцедуры
```

Подключаемый список полей может быть кастомизирован. См. описание в функции `КонструкторФормул.ПараметрыДобавленияСпискаПолей`.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Контактная информация

Подсистема **Контактная информация** позволяет расширять состав реквизитов объекта произвольным набором реквизитов контактной информации. Реквизиты контактной информации могут быть как предопределенными (адреса, телефоны, факс, e-mail, www), так и пользовательскими (например, «телефон с 18:00 до 24:00»). Справочник со встроенной контактной информацией называется владельцем контактной информации.

> **Примечание**
>
> В российской версии библиотеки вид контактной информации **Адрес** использует возможности подсистемы **Адресный классификатор**. Основной режим использования подсистемы предполагает, что в конфигурацию также переносится подсистема **Адресный классификатор**.
>
> Загрузка классификатора стран мира может производиться как автоматически с портала 1С:ИТС, так и администратором из указанного файла (при наличии подсистемы **Работа с классификаторами** библиотеки **1С:Библиотека интернет-поддержки**.

Настройка

После выполнения переноса объектов библиотеки необходимо включить в командный интерфейс конфигурации справочники:

- `ВидыКонтактнойИнформации`,
- `СтраныМира`.

Настройка объектов подсистемы сводится к созданию в справочнике `ВидыКонтактнойИнформации` предопределенных элементов и вызову обработчика обновления информационной базы, использующего процедуру `УстановитьСвойстваВидаКонтактнойИнформации` общего модуля `УправлениеКонтактнойИнформацией`, для установки значения реквизитов этих элементов.

1. Принять решение по поводу состава объектов – владельцев контактной информации и видов контактной информации у каждого такого объекта. Например, у справочника **Контактные лица** необходимо разместить телефон и адрес, а у справочника **Контрагент** – адрес электронной почты.
2. Добавить ссылки на объекты-владельцы (кроме документов) в состав определяемого типа `ВладелецКонтактнойИнформации`.
3. Добавить ссылки на документы-владельцы в подписку на событие `ЗаполнитьКонтактнуюИнформациюДокумента`.
4. Добавить описание видов контактной информации в процедуру `ПриНачальномЗаполненииЭлементов` общего модуля `УправлениеКонтактнойИнформациейПереопределяемый`.
   - На первом уровне вводятся группы: для каждого вида объекта – владельца контактной информации – своя группа. У группы должно быть задано имя в формате `Справочник<ИмяСправочника>` или `Документ<ИмяДокумента>`. Например, для справочника контактные лиц – `СправочникКонтактныеЛица`. Наименование группы рекомендуется задавать в виде: **Контактная информация справочника "Контактные лица"**.
   - На втором уровне вводятся элементы – виды контактной информации, список которых определяет разработчик конфигурации. На втором уровне могут быть также заведены группы, если предполагается хранение контактной информации для табличной части объекта. В этом случае у группы должно быть задано имя в формате `Документ<ИмяСправочника><ИмяТабличнойЧасти>`.
   - На третьем уровне вводятся элементы – виды контактной информации для табличной части объекта. Список видов контактной информации определяет ответственный за конфигурацию.

Пример:

```
Процедура ПриНачальномЗаполненииВидовКонтактнойИнформации(КодыЯзыков, Элементы, ТабличныеЧасти) Экспорт

    // Контактная информация справочника "Контактные лица"
    Элемент = Элементы.Добавить();
    Элемент.ИмяПредопределенногоВида = "СправочникКонтактныеЛица";
    Элемент.ЭтоГруппа = Истина;
    Элемент.Наименование = НСтр("ru = 'Контактная информация справочника ""Контактные лица""'", ОбщегоНазначения.КодОсновногоЯзыка());

    // Адрес контактного лица
    Элемент = Элементы.Добавить();
    Элемент.Родитель = "СправочникКонтактныеЛица";
    Элемент.ИмяГруппы = "СправочникКонтактныеЛица";
    Элемент.Тип = Перечисления.ТипыКонтактнойИнформации.Адрес;
    Элемент.ИмяПредопределенногоВида = "АдресКонтактногоЛица";
    Элемент.ИдентификаторДляФормул = "АдресКонтактногоЛица";
    Элемент.ВидРедактирования = "ПолеВвода";
    Элемент.ВключатьСтрануВПредставление = Ложь;
    Элемент.ХранитьИсториюИзменений = Ложь;
    Элемент.ОтображатьВсегда = Истина;
    Элемент.Наименование = НСтр("ru = 'Адрес'", ОбщегоНазначения.КодОсновногоЯзыка());

КонецПроцедуры
```

Для мультиязычных конфигураций стандартный реквизит `Наименование` следует заполнять с использованием процедуры `ЗаполнитьМультиязычныйРеквизит` общего модуля `МультиязычностьСервер`.

```
МультиязычностьСервер.ЗаполнитьМультиязычныйРеквизит(Элемент, "Наименование",
    "ru = 'Адрес'; en = 'Address'", КодыЯзыков); // @НСтр-1
```

Для обновления значений реквизитов видов контактной информации необходимо использовать оперативный обработчик обновления. Примеры использования обработчика `АктуализироватьВидыКонтактнойИнформации` см. в общем модуле `_ДемоОбновлениеИнформационнойБазыБСП` демонстрационной конфигурации.

> **Примечание**
> Подробнее о разработке предопределенных элементов см. в разделе **Начальное заполнение данных** подсистемы **Обновление версии ИБ**

Настройка объектов – владельцев контактной информации

1. У объекта – владельца контактной информации необходимо создать табличную часть `КонтактнаяИнформация` со следующими обязательными реквизитами:

Имя	Тип значения	Подсказка
`Тип`	`ПеречислениеСсылка.ТипыКонтактнойИнформации`	Тип контактной информации (телефон, адрес и т. п.)
`Вид`	`СправочникСсылка.ВидыКонтактнойИнформации`	Вид контактной информации
`Представление`	`Строка` (500)	Представление контактной информации для отображения в формах
`Значение`	`Строка` неограниченной длины	Служебное поле для хранения контактной информации
`Страна`	`Строка` (100)	Страна (заполняется для адреса)
`Регион`	`Строка` (50)	Регион (заполняется для адреса)
`Город`	`Строка` (50)	Город (заполняется для адреса)
`АдресЭП`	`Строка` (100)	Адрес электронной почты
`ДоменноеИмяСервера`	`Строка` (100)	Доменное имя сервера электронной почты или веб-страницы
`НомерТелефона`	`Строка` (20)	Полный номер телефона
`НомерТелефонаБезКодов`	`Строка` (20)	Номер телефона без кодов и добавочного номера
`ЗначенияПолей`	`Строка` неограниченной длины	Служебное поле для хранения контактной информации для поддержания обратной совместимости при обмене данными

Для реквизитов `Вид` и `Тип` необходимо включить индексирование. Для всех строковых реквизитов ограниченной длины следует устанавливать индексы по необходимости. Т. е. индексирование должно быть включено только для тех реквизитов, которые используются для выборки данных. Для всех реквизитов, кроме реквизита `Представление`, следует выключить полнотекстовый поиск. Кроме того, для задействования дополнительных возможностей подсистемы список реквизитов табличной части `КонтактнаяИнформация` может также включать в себя:

Имя	Тип значения	Подсказка
`ИдентификаторСтрокиТабличнойЧасти` [1]	`Число` (7,0)	Идентификатор строки табличной части
`ВидДляСписка` [2]	`СправочникСсылка.ВидыКонтактнойИнформации`	Вид контактной информации, отображаемый в списках
`ДействуетС` [3]	`Дата`	Дата, с которой действует запись контактной информации

> [1] Если предполагается хранение контактной информации для табличной части объекта. В этом случае необходимо также добавить этот же реквизит в табличную часть объекта. Пример реализации см. в документе `_ДемоЗаказПокупателя` демонстрационной конфигурации.
>
> [2] Если предполагается вывод полей контактной информации в списках и отчетах.
>
> [3] Если предполагается хранение истории изменения контактной информации.

1. Изменить форму элемента:

- Если предполагается хранение контактной информации для объекта, то в форме элемента создать группу или страницу с именем `ГруппаКонтактнаяИнформация` и задать ей заголовок – **Адреса, телефоны** (или другой).
- Если предполагается хранение контактной информации для табличной части объекта, то создать группу колонок с именем `<ИмяТабличнойЧасти>ГруппаКонтактнаяИнформация` в нужной табличной части объекта. Например, для табличной части `Партнеры` группа контактной информации должна иметь вид `ПартнерыГруппаКонтактнаяИнформация`.

1. Изменить модуль формы элемента.

- В обработчики событий формы `ПриСозданииНаСервере`, `ПриЧтенииНаСервере`, `ОбработкаПроверкиЗаполненияНаСервере` и `ПередЗаписьюНаСервере` вставить строки из соответствующих процедур, располагающихся ниже. Если предполагается хранение контактной информации для табличной части объекта, то в обработчик события формы `ПослеЗаписиНаСервере` вставить строки из соответствующей процедуры ниже. Добавляемые строки должны располагаться после проверок, которые могут вызвать отказ в событии.

```
#Область ОбработчикиСобытийФормы
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    ДополнительныеПараметры = УправлениеКонтактнойИнформацией.ПараметрыКонтактнойИнформации();
    ДополнительныеПараметры.Вставить("ИмяЭлементаДляРазмещения", "ГруппаКонтактнаяИнформация");
    УправлениеКонтактнойИнформацией.ПриСозданииНаСервере(ЭтотОбъект, Объект, ДополнительныеПараметры);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация

КонецПроцедуры
&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация

КонецПроцедуры
&НаСервере
Процедура ОбработкаПроверкиЗаполненияНаСервере(Отказ, ПроверяемыеРеквизиты)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ОбработкаПроверкиЗаполненияНаСервере(ЭтотОбъект, Объект, Отказ);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация
КонецПроцедуры
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ПередЗаписьюНаСервере(ЭтотОбъект, ТекущийОбъект);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация
КонецПроцедуры
&НаСервере
Процедура ПослеЗаписиНаСервере(ТекущийОбъект, ПараметрыЗаписи)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ПослеЗаписиНаСервере(ЭтотОбъект, ТекущийОбъект);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация

КонецПроцедуры
#КонецОбласти
```

Здесь `ГруппаКонтактнаяИнформация`, ссылка на элемент формы – контейнер, в который будут добавляться элементы для редактирования контактной информации. Например:

```
ДополнительныеПараметры = УправлениеКонтактнойИнформацией.ПараметрыКонтактнойИнформации()
ДополнительныеПараметры.Вставить("ИмяЭлементаДляРазмещения", "ГруппаКонтактнаяИнформация");
УправлениеКонтактнойИнформацией.ПриСозданииНаСервере(ЭтотОбъект, Объект, ДополнительныеПараметры);
```

- В вызове процедуры `УправлениеКонтактнойИнформацией.ПриСозданииНаСервере` в качестве третьего параметра нужно указать дополнительные параметры список, которых определен в функции-конструкторе `УправлениеКонтактнойИнформацией.` `ПараметрыКонтактнойИнформации.` Например, свойство `ИмяЭлементаДляРазмещения` содержит имя элемента, в котором будут располагаться создаваемые поля ввода, а свойство `ПоложениеЗаголовкаКИ` позволяет задать положение заголовка элементов контактной информации. Заголовки могут располагаться сверху и слева (значения параметра `ПоложениеЗаголовкаЭлементаФормы.Лево` и `ПоложениеЗаголовкаЭлементаФормы.Верх` соответственно). Подробнее см. описание процедуры `ПараметрыКонтактнойИнформации` общего модуля `УправлениеКонтактнойИнформацией`.

1. Перенести блок процедур контактной информации, расположенный ниже, в модуль формы объекта – владельца контактной информации.

```
// СтандартныеПодсистемы.КонтактнаяИнформация
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияПриИзменении(Элемент)
        УправлениеКонтактнойИнформациейКлиент.НачатьИзменение(ЭтотОбъект, Элемент);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)
    УправлениеКонтактнойИнформациейКлиент.НачатьВыбор(ЭтотОбъект, Элемент, , СтандартнаяОбработка);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияПриНажатии(Элемент, СтандартнаяОбработка)
    УправлениеКонтактнойИнформациейКлиент.НачатьВыбор(ЭтотОбъект, Элемент, , СтандартнаяОбработка);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияОчистка(Элемент, СтандартнаяОбработка)
    УправлениеКонтактнойИнформациейКлиент.НачатьОчистку(ЭтотОбъект, Элемент.Имя);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияВыполнитьКоманду(Команда)
    УправлениеКонтактнойИнформациейКлиент.НачатьВыполнениеКоманды(ЭтотОбъект, Команда.Имя);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияАвтоПодбор(Элемент, Текст, ДанныеВыбора, ПараметрыПолученияДанных, Ожидание, СтандартнаяОбработк

    УправлениеКонтактнойИнформациейКлиент.АвтоПодборАдреса(Элемент, Текст, ДанныеВыбора, ПараметрыПолученияДанных, Ожидание, СтандартнаяОбр

КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияОбработкаВыбора(Элемент, ВыбранноеЗначение, СтандартнаяОбработка)

    УправлениеКонтактнойИнформациейКлиент.ОбработкаВыбора(ЭтотОбъект, ВыбранноеЗначение, Элемент.Имя, СтандартнаяОбработка);

КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_КонтактнаяИнформацияОбработкаНавигационнойСсылки(Элемент, НавигационнаяСсылкаФорматированнойСтроки, СтандартнаяОбраб
    УправлениеКонтактнойИнформациейКлиент.НачатьОбработкуНавигационнойСсылки(ЭтотОбъект, Элемент, НавигационнаяСсылкаФорматированнойСтроки,
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ПродолжитьОбновлениеКонтактнойИнформации(Результат, ДополнительныеПараметры) Экспорт
    ОбновитьКонтактнуюИнформацию(Результат);
КонецПроцедуры
&НаСервере
Процедура ОбновитьКонтактнуюИнформацию(Результат)
    УправлениеКонтактнойИнформацией.ОбновитьКонтактнуюИнформацию(ЭтотОбъект, Объект, Результат);
КонецПроцедуры
// Конец СтандартныеПодсистемы.КонтактнаяИнформация
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Контактная информация** следует использовать роли, перечисленные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Чтение видов контактной информации и списка стран.
2.	`ДобавлениеИзменениеВидовКонтактнойИнформации`. Ведение списка стран и видов контактной информации.
3.	`ПолучениеОбновленийКлассификаторов` (из библиотеки **1С:Библиотека интернет-поддержки**) Обновление классификатора стран мира с портала 1С:ИТС.
4.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Ведение видов контактной информации, списка стран, а также удаление помеченных на удаление объектов подсистемы.

Для просмотра и редактирования контактной информации конкретных объектов требуются роли для просмотра и редактирования самих объектов-владельцев контактной информации.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Ответственный за нормативно-справочную информацию (НСИ): ведение видов контактной информации, списка стран, а также всей остальной НСИ	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ПолучениеОбновленийКлассификаторов` (из библиотеки **1С:Библиотека интернет-поддержки**), • `ДобавлениеИзменениеВидовКонтактнойИнформации`
3.	Менеджер по закупкам: ведение списка номенклатуры, ведение списка стран мира (для ввода стран происхождения товаров)	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеВидовКонтактнойИнформации`, • `ДобавлениеИзменениеНоменклатуры`.

Особые случаи внедрения подсистемы

Вывод полей контактной информации в списках и отчетах

Для вывода контактной информации в динамических списках и отчетах необходимо выполнить следующую настройку. В объекте – владельце контактной информации в табличную часть `КонтактнаяИнформация` добавить реквизит `ВидДляСписка` с типом `СправочникСсылка.ВидыКонтактнойИнформации`, затем в форме **Дополнительные характеристики объекта данных** добавить дополнительную характеристику:

Виды характеристик	Поле ключа	Поле отбора видов	Значение отбора видов
`Справочник.ВидыКонтактнойИнформации`	`Ссылка`	`ИмяГруппы`	Имя группа справочника `ВидыКонтактнойИнформации`
Значения характеристик	**Поле объекта**	**Поле вида**	**Поле значения**
`Справочник.[Имя объекта с контактной информацией].ТабличнаяЧасть.КонтактнаяИнформация`	`Ссылка`	`ВидДляСписка`	`Представление`

где `Предопределенная` группа (элемент) справочника `ВидыКонтактнойИнформации` – предопределенная группа(элемент), созданная на этапе формирования видов контактной информации. Для отображения контактной информации в списках – в форме списка в меню **Еще** выбрать пункт **Изменить форму**. В таблице **Элементы формы** выбрать реквизит **Ссылка**, нажать кнопку **Добавить поля** в командной панели формы и далее в открывшемся списке полей выбрать нужные поля контактной информации для отображения.

См. пример реализации в справочнике `_ДемоКонтрагенты` демонстрационной конфигурации.

Вывод картинок у полей контактной информации

Для вывода информационных картинок слева от заголовков полей контактной информации нужно установить значение `Истина` у параметра `ОтображатьИконки` в процедуре `ПриОпределенииНастроек` общего модуля `УправлениеКонтактнойИнформациейПереопределяемый`.

Переопределение команд контактной информации

В некоторых случаях может потребоваться переопределить стандартные команды в полях контактной информации, отображаемых в справочниках и документах: написать письмо, отправить SMS, позвонить и другие. Для этого нужно использовать параметр `ОписаниеКоманд` процедуры `ПриОпределенииНастроек` общего модуля `УправлениеКонтактнойИнформациейПереопределяемый`. С его помощью возможно изменить, назначить свой обработчик, добавить или удалить лишние стандартные команды в полях контактной информации. Пример переопределения команд см. в демонстрационной конфигурации в процедуре `ПриОпределенииНастроек` общего модуля `УправлениеКонтактнойИнформациейПереопределяемый`.

Создание статических элементов управления

Создание статических реквизитов контактной информации позволяет:

- добавлять элементы контактной информации в различные части формы (например, вынести самые важные виды контактной информации в шапку);
- изменять внешний вид и свойства элементов контактной информации без внесения изменений в типовую конфигурацию;
- повысить скорость открытия формы – владельца контактной информации.

Для создания статических элементов контактной информации необходимо:

- принять решение о видах контактной информации, элементы управления которых должны создаваться вручную;
- в обработчике `ПриСозданииНаСервере` перечислить выбранные виды в поле `РазмещеныНаФорме` структуры `ДополнительныеПараметры` в качестве третьего параметра при вызове процедуры `УправлениеКонтактнойИнформацией.ПриСозданииНаСервере`;
- добавить в форму реквизит `ПараметрыКонтактнойИнформации`, тип – произвольный;
- добавить в форму реквизит `КонтактнаяИнформацияОписаниеДополнительныхРеквизитов`, тип – таблица значений с колонками:
  - `ИмяРеквизита` – `Строка`;
  - `ИмяРеквизитаКомментарий` – `Строка`;
  - `Вид` – `СправочникСсылка.ВидыКонтактнойИнформации`;
  - `Тип` – `ПеречислениеСсылка.ТипыКонтактнойИнформации`;
  - `Значение` - `Строка`;
  - `Представление` – `Строка`;
  - `Комментарий` – `Строка`;
  - `ЭтоРеквизитТабличнойЧасти` – `Булево`;
  - `ЭтоИсторическаяКонтактнаяИнформация` – `Булево`;
  - `МеждународныйФорматАдреса` – `Булево`;
  - `ИмяЭлементаДляРазмещения` – `Строка`;
  - `ЗначенияПолей` – `СписокЗначений`, `Строка`.
- для каждого вида контактной информации добавить реквизит с именем вида `КонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`;
- для типов: `Skype`, `АдресЭлектроннойПочты`, `ВебСтраница`, `Телефон`, `Факс`, нужно добавить реквизит с именем вида `КомментарийКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`;
- добавить на форму элементы:
  - для каждого вида контактной информации добавить на форму группу с именем `ГруппаКонтактнаяИнформацияПоле <ИмяВидаКонтактнойИнформации>`. Свойства: `Отображение` – `Нет`, `Группировка` – `Горизонтальная`, `ОтображатьЗаголовок` – `Ложь`;
  - для типов: `Skype`, `АдресЭлектроннойПочты`, `ВебСтраница`, `Телефон`, `Факс`, нужно добавить группу с именем `ГруппаКомментарийКонтактнаяИнформацияПоле <ИмяВидаКонтактнойИнформации>`. Свойства: `Отображение` – `Нет`, `Группировка` – `Горизонтальная`, `ОтображатьЗаголовок` – `Ложь`;
  - для типов: `Skype`, `АдресЭлектроннойПочты`, `ВебСтраница`, `Телефон`, `Факс`: В созданную группу с именем `ГруппаКомментарийКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>` нужно перетащить ранее созданный реквизит с именем `КомментарийКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`, отключив отображение заголовка и установив подсказку ввода **Примечание**.
  - для типов `Skype`, `Адрес электронной почты`, `ВебСтраница`, `Телефон`, `Факс`: в созданную группу с именем `ГруппаКонтактнаяИнформацияПоле <ИмяВидаКонтактнойИнформации>` перетащить ранее созданный реквизит с именем вида `КонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`, установив соответствующее отображение заголовка (`Лево` или `Верх`) в зависимости от положения заголовка, передаваемого в процедуру `УправлениеКонтактнойИнформацией.ПриСозданииНаСервере`;
  - для типов `Адрес` и `Другое`: в созданную группу именем `ГруппаКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>` перетащить ранее созданный реквизит с именем вида `КонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`, установив соответствующее отображение заголовка (`Лево` или `Верх`) в зависимости от положения заголовка, передаваемого в процедуру `УправлениеКонтактнойИнформацией.ПриСозданииНаСервере`;
  - для добавления действия необходимо создать команду с именем вида `КомандаКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`, отображение `Картинка`. Затем эту команду необходимо перетащить в ранее созданную группу `ГруппаКонтактнаяИнформацияПоле<ИмяВидаКонтактнойИнформации>`. Картинка команды, заголовок и подсказка устанавливаются в зависимости от типа контактной информации:
    - для типа `Адрес` картинка `МенюДополнительныеФункции`;
    - для типа `ВебСтраница` картинка `КонтактнаяИнформацияПерейтиПоСсылке`, заголовок **Перейти**, подсказка **Перейти по ссылке**;
    - для типа `АдресЭлектроннойПочты` действие устанавливается только при наличии подсистемы РаботаСПочтовымиСообщениями. При этом устанавливается картинка `ОтправитьЭлектронноеПисьмо`, заголовок **Написать письмо**, подсказка **Написать письмо**;
    - для типа `Телефон` при наличии подсистемы ОтправкаSMS используется картинка `МенюДополнительныеФункции`, заголовок **Позвонить или отправить SMS**, подсказка **Позвонить или отправить SMS**. Если подсистемы ОтправкаSMS нет, то используется картинка `Позвонить`, заголовок **Позвонить**, подсказка **Позвонить по телефону**;
    - для типа `Skype` необходимо установить картинку `МенюДополнительныеФункции`, заголовок и подсказку **Skype**.

Картинки команд, заголовки и подсказки следует устанавливать в соответствии с реализацией процедуры `ПриОпределенииНастроек` общего модуля `УправлениеКонтактнойИнформациейПереопределяемый` (если ее реализация определена).

Виды контактной информации, добавляемые в форму вручную, рекомендуется сделать недоступными для редактирования пользователем. Это можно сделать в обработчике обновления при помощи процедуры `УстановитьСвойстваВидаКонтактнойИнформации` из общего модуля `УправлениеКонтактнойИнформацией`. За недоступность редактирования пользователем отвечает свойство `ЗапретитьРедактированиеПользователем`.

Пример использования см. в форме элемента справочника `_ДемоКонтрагенты` демонстрационной конфигурации.

Отложенная инициализация элементов контактной информации

Если элементы контактной информации (или их часть) располагаются на отдельной закладке и работа с ними не является основным сценарием работы с формой, то можно увеличить скорость открытия формы путем переноса создания элементов контактной информации с момента открытия формы на момент перехода на закладку контактной информации.

Использование отложенной инициализации также накладывает следующие ограничения:

- невозможно перетаскивать элементы со страницы контактной информации,
- в некоторых случаях возможны «скачки» формы.

Для использования отложенной инициализации необходимо:

- добавить в форму реквизит `ПараметрыКонтактнойИнформации`, тип – произвольный;
- добавить в форму реквизит `КонтактнаяИнформацияОписаниеДополнительныхРеквизитов`, тип – таблица значений. Описание колонок и их типов см. в предыдущем разделе;
- в обработчик `УправлениеКонтактнойИнформацией.ПриСозданииНаСервере` передать в качестве шестого параметра (`ОтложеннаяИнициализация`) значение `Истина`;
- для страницы контактной информации установить свойство `РазрешитьИзменениеСостава` в значение `Ложь`;
- для группы страниц, содержащей страницу контактной информации, добавить обработчик `ПриСменеСтраницы`, в который добавить код:

```
// СтандартныеПодсистемы.КонтактнаяИнформация
Если ПараметрыКонтактнойИнформации.Свойство(ТекущаяСтраница.Имя)
    И НЕ ПараметрыКонтактнойИнформации[ТекущаяСтраница.Имя].ВыполненаОтложеннаяИнициализация Тогда
    КонтактнаяИнформацияПриСменеСтраницы();
КонецЕсли;
// Конец СтандартныеПодсистемы.КонтактнаяИнформация
```

- в блок процедур контактной информации добавить процедуру:

```
&НаСервере
Процедура КонтактнаяИнформацияПриСменеСтраницы()
    УправлениеКонтактнойИнформацией.ВыполнитьОтложеннуюИнициализацию(ЭтотОбъект, Объект);
КонецПроцедуры
```

- если на странице контактной информации нет статических элементов, то добавить декорацию с именем `ПустаяДекорацияКонтактнаяИнформация`.

Пример использования см. в форме элемента справочника `_ДемоКонтрагенты` демонстрационной конфигурации.

Хранение истории изменений контактной информации

В случае если для ведения учета важно хранить не только текущие значения полей контактной информации, но и их историю, а также получать значение на указанную дату, необходимо настроить хранение истории изменения контактной информации:

1. Принять решение по поводу состава объектов – владельцев контактной информации и видов контактной информации, у которых необходимо хранить историю изменения контактной информации. Например, у справочника `Организации` это может быть юридический адрес.
2. Добавить в табличную часть `КонтактнаяИнформация` этих объектов реквизит `ДействуетС` с типом `Дата`.
3. Написать обработчик обновления, устанавливающий флажок необходимости хранения история изменений для вида контактной информации. Пример кода обработчика:

```
ПараметрыВида = УправлениеКонтактнойИнформацией.ПараметрыВидаКонтактнойИнформации("Адрес");
ПараметрыВида.ХранитьИсториюИзменений = Истина;
УправлениеКонтактнойИнформацией.УстановитьСвойстваВидаКонтактнойИнформации(ПараметрыВида);
```

Для получения контактной информации на заданную дату необходимо в процедурах и функциях `КонтактнаяИнформацияОбъекта`, `КонтактнаяИнформацияОбъектов`, `СоздатьВТКонтактнаяИнформация`, `ТаблицаКонтактнойИнформацииОбъекта` указывать параметр `Дата` (например, дату документа в коде печатных форм). Для записи контактной информации на заданную дату следует использовать процедуры `ЗаполнитьКонтактнуюИнформациюОбъектов` и `ЗаполнитьКонтактнуюИнформациюОбъекта` аналогичным образом. При включенной истории хранения изменений не допускается возможность ввода нескольких значений контактной информации. Более подробную информацию см. в комментариях к процедурам.

Пример реализации см. в демонстрационной конфигурации в справочнике `_ДемоОрганизации` для вида контактной информации **Юридический адрес**.

Заполнение и редактирование контактной информации одного объекта в форме другого объекта

Имеется возможность выводить контактную информацию объекта в отдельной форме или форме другого объекта. Такая возможность позволяет редактировать контактную информацию объекта без необходимости открытия формы владельца контактной информации.

Например, если необходимо для вывода к уже ранее размещенной контактной информации элемента справочника **Демо: Контактные лица партнеров** выводить контактную информацию реквизита `ФизическоеЛицо`, ссылающегося на элемент справочника **Демо: Физические лица**, то следует сделать следующее:

- создать реквизит формы `ФизическоеЛицо` с типом `СправочникОбъект._ДемоФизическиеЛица`, содержащий объект владельца редактируемой контактной информацией;
- на форме создать группу или страницу с именем `ГруппаКонтактнаяИнформацияФизическоеЛицо` – контейнер, в который будут добавляться элементы формы для редактирования контактной информации физического лица;
- в обработчиках событий формы `ПриСозданииНаСервере`, `ПриЧтенииНаСервере` и `ПередЗаписьюНаСервере` для заполнения и сохранения контактной информации физического лица добавить аналогичные вызовы, где в качестве параметров указать реквизит формы `ФизическоеЛицо` и группу размещения контактной информации:

```
#Область ОбработчикиСобытийФормы
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ПриСозданииНаСервере(ЭтотОбъект, Объект,, ПоложениеЗаголовкаЭлементаФормы.Лево);
    УправлениеКонтактнойИнформацией.ПриСозданииНаСервере(ЭтотОбъект, ФизическоеЛицо, "ГруппаКонтактнаяИнформацияФизическогоЛица", Положение

    // Конец СтандартныеПодсистемы.КонтактнаяИнформация

КонецПроцедуры
&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)

    // СтандартныеПодсистемы.КонтактнаяИнформация
    Если ЗначениеЗаполнено(ТекущийОбъект.ФизическоеЛицо) Тогда
        ЗначениеВРеквизитФормы(ТекущийОбъект.ФизическоеЛицо.ПолучитьОбъект(), "ФизическоеЛицо");
    КонецЕсли;
    УправлениеКонтактнойИнформацией.ПриЧтенииНаСервере(ЭтотОбъект, ФизическоеЛицо, "ГруппаКонтактнаяИнформацияФизическогоЛица");
    УправлениеКонтактнойИнформацией.ПриЧтенииНаСервере(ЭтотОбъект, Объект);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация

КонецПроцедуры
&НаСервере
Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
    // СтандартныеПодсистемы.КонтактнаяИнформация
    УправлениеКонтактнойИнформацией.ПередЗаписьюНаСервере(ЭтотОбъект, ТекущийОбъект);
    УправлениеКонтактнойИнформацией.ПередЗаписьюНаСервере(ЭтотОбъект, ФизическоеЛицоОбъект);
    // Конец СтандартныеПодсистемы.КонтактнаяИнформация
КонецПроцедуры
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
    ФизическоеЛицоОбъект = РеквизитФормыВЗначение("ФизическоеЛицо");
    ФизическоеЛицоОбъект.Записать();
    ЗначениеВРеквизитФормы(ФизическоеЛицоОбъект, "ФизическоеЛицо");
КонецПроцедуры
#КонецОбласти
```

- добавить код, блокирующий для изменения элемент справочника **Демо: Физические лица**, контактная информация которого выведена на форму:
  - для этого в обработчики события `ПриОткрытии` подключить обработчик, проверяющий необходимость блокировки объекта физического лица:

    ```
    &НаКлиенте
    Процедура ПриОткрытии(Отказ)
        ПодключитьОбработчикОжидания("ПроверитьНеобходимостьБлокировкиФизическогоЛица", 1, Ложь);
    КонецПроцедуры
    ```

  - и процедуру, вызываемую этим обработчиком для попытки блокировки элемента справочника физического лица, если информация на форме была изменена:

    ```
    &НаКлиенте
    Процедура ПроверитьНеобходимостьБлокировкиФизическогоЛица()
        Если Модифицированность И Не ФизическоеЛицо.Ссылка.Пустая() Тогда
            Если ЗаблокироватьФизическоеЛицоПриРедактированииНаСервере(ФизическоеЛицо.Ссылка, УникальныйИдентификатор) Тогда
                ОтключитьОбработчикОжидания("ПроверитьНеобходимостьБлокировкиФизическогоЛица");
            Иначе
                Прочитать();
                ВызватьИсключение НСтр("ru = 'Данные контактного лица не могут быть записаны, т.к. личные данные физического лица не досту
                |Возможно, эти данные физического лица редактируются другим пользователем.'");
            КонецЕсли;
        КонецЕсли;
    КонецПроцедуры

    &НаСервереБезКонтекста
    Функция ЗаблокироватьФизическоеЛицоПриРедактированииНаСервере(ФизическоеЛицо, ФормаУникальныйИдентификатор)

        Попытка
            ЗаблокироватьДанныеДляРедактирования(ФизическоеЛицо, ФизическоеЛицо.ВерсияДанных, ФормаУникальныйИдентификатор);
        Исключение
            Возврат Ложь;
        КонецПопытки;

        Возврат Истина;

    КонецФункции
    ```

Пример использования см. в форме элемента справочника `_ДемоКонтактныеЛицаПартнеров` демонстрационной конфигурации.

Добавление произвольных реквизитов с контактной информацией

Имеется возможность хранить значения полей контактной информации не только в стандартной табличной части `КонтактнаяИнформация`, но и в произвольных реквизитах произвольных объектов конфигурации. Такая возможность может быть востребована для добавления к объектам отдельных полей с контактной информацией.

Например, необходимо хранить данные адреса доставки в документе `ЗаказПокупателя`. При этом в форме документа должно быть поле **Адрес доставки**, редактируемое с помощью стандартной формы ввода адреса, и поля **Дата доставки**, **Метро**, **Комментарий**. При этом в заказе покупателя не требуются остальные сервисы подсистемы по заведению других видов контактной информации.

Для решения этой задачи необходимо:

1. Добавить к документу `ЗаказПокупателя` основные реквизиты:
   - `АдресДоставки` типа `Строка` неограниченной длины. Этот реквизит будет хранить служебные значения контактной информации адреса доставки;
   - `ДатаДоставки` типа `Дата` для хранения даты доставки;
   - `Метро`. Ссылка на справочник `СтанцииМетро`.
2. Добавить к документу `ЗаказПокупателя` вспомогательные реквизиты, данные в которых будут вычисляться при записи документа по данным реквизита `АдресДоставки`. Эти реквизиты предназначены для оперативного поиска документов по данным контактной информации, вывода данных на печать и т. п.
   - `АдресДоставкиСтрокой` – строка длиной 200 символов;
   - `ГородАдресаДоставки` – строка длиной 100 символов.
3. Поместить в форму вспомогательный реквизит `ВидКонтактнойИнформацииАдресаДоставки` для контроля работы с адресом и инициализировать его при создании формы:

   ```
   ВидКонтактнойИнформацииАдресаДоставки = Новый Структура;
   ВидКонтактнойИнформацииАдресаДоставки.Вставить("Тип", Перечисления.ТипыКонтактнойИнформации.Адрес);
   ВидКонтактнойИнформацииАдресаДоставки.Вставить("ТолькоНациональныйАдрес", Ложь);
   ВидКонтактнойИнформацииАдресаДоставки.Вставить("ВключатьСтрануВПредставление", Ложь);
   ВидКонтактнойИнформацииАдресаДоставки.Вставить("СкрыватьНеактуальныеАдреса", Ложь);
   ```

   Эта структура заменяет собой соответствующий элемент справочника `ВидыКонтактнойИнформации`.
4. Поместить в форму документа вспомогательный строковый реквизит `ПредставлениеАдресаДоставки`. Инициализировать значение реквизита при создании формы:

   ```
   ПредставлениеАдресаДоставки = УправлениеКонтактнойИнформацией.ПредставлениеКонтактнойИнформации(Объект.АдресДоставки);
   ```

   - создать соответствующее вспомогательному реквизиту поле ввода, установить у него флажок `КнопкаВыбора` в `Истина`;
   - определить обработчики событий для этого элемента:

```
&НаКлиенте
Процедура ПредставлениеАдресаДоставкиПриИзменении(Элемент)

    Текст = Элемент.ТекстРедактирования;
    Если ПустаяСтрока(Текст) Тогда
        ПредставлениеАдресаДоставки = "";
        КомментарийАдресаДоставки   = "";
        Объект.АдресДоставки        = "";
        Возврат;
    КонецЕсли;
    ПредставлениеАдресаДоставки = Текст;
    Объект.АдресДоставки = ЗначенияПолейКонтактнойИнформацииСервер(Текст,
        ВидКонтактнойИнформацииАдресаДоставки, КомментарийАдресаДоставки);
КонецПроцедуры
&НаКлиенте
Процедура ПредставлениеАдресаДоставкиНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)

    Если Элемент.ТекстРедактирования<>ПредставлениеАдресаДоставки Тогда
        ПредставлениеАдресаДоставки = Элемент.ТекстРедактирования;
        Объект.АдресДоставки = "";
    КонецЕсли;

    ПараметрыОткрытия = Новый Структура;
    ПараметрыОткрытия.Вставить("ВидКонтактнойИнформации", ВидКонтактнойИнформацииАдресаДоставки);
    ПараметрыОткрытия.Вставить("Значение",      Объект.АдресДоставки);
    ПараметрыОткрытия.Вставить("Представление", ПредставлениеАдресаДоставки);
    ПараметрыОткрытия.Вставить("Комментарий",   КомментарийАдресаДоставки);
    ПараметрыОткрытия.Вставить("Заголовок", НСтр("ru='Адрес доставки'"));

    УправлениеКонтактнойИнформациейКлиент.ОткрытьФормуКонтактнойИнформации(ПараметрыОткрытия, Элемент);
КонецПроцедуры
&НаКлиенте
Процедура ПредставлениеАдресаДоставкиОчистка(Элемент, СтандартнаяОбработка)
    ПредставлениеАдресаДоставки = "";
    КомментарийАдресаДоставки   = "";
    Объект.АдресДоставки        = "";
КонецПроцедуры
&НаКлиенте
Процедура ПредставлениеАдресаДоставкиОбработкаВыбора(Элемент, ВыбранноеЗначение, СтандартнаяОбработка)
    СтандартнаяОбработка = Ложь;
    Если ТипЗнч(ВыбранноеЗначение)<>Тип("Структура") Тогда
        // Отказ от выбора, данные неизменны
        Возврат;
    КонецЕсли;

    ПредставлениеАдресаДоставки = ВыбранноеЗначение.Представление;
    КомментарийАдресаДоставки   = ВыбранноеЗначение.Комментарий;
    Объект.АдресДоставки        = ВыбранноеЗначение.КонтактнаяИнформация;
КонецПроцедуры
```

Следует заметить, что если функция ввода адреса строкой не требуется, то можно обойтись без этого вспомогательного поля ввода, заменив его, например, на гиперссылку. Текст гиперссылки при этом будет представлением адреса, а обработчик нажатия аналогичен обработчику `НачалоВыбора`.

5. Поместить в форму вспомогательный реквизит `КомментарийАдресаДоставки` для редактирования комментария, не попадающего в основное представление адреса. Инициализировать его при создании формы:

```
КомментарийАдресаДоставки =
УправлениеКонтактнойИнформацией.КомментарийКонтактнойИнформации(Объект.АдресДоставки);
```

6. Поместить в форму поле ввода и связать его с реквизитом `КомментарийАдресаДоставки`. Определить следующие обработчики изменений:

```
&НаКлиенте
Процедура КомментарийАдресаДоставкиПриИзменении(Элемент)
    ЗаполнитьКомментарийАдресаДоставкиСервер();
КонецПроцедуры

&НаСервере
Процедура ЗаполнитьКомментарийАдресаДоставкиСервер()
    Если ПустаяСтрока(Объект.АдресДоставки) Тогда

        Объект.АдресДоставки = ЗначенияПолейКонтактнойИнформацииСервер(ПредставлениеАдресаДоставки,
                ВидКонтактнойИнформацииАдресаДоставки, КомментарийАдресаДоставки);
        Возврат;
    КонецЕсли;

    УправлениеКонтактнойИнформацией.КомментарийКонтактнойИнформации(Объект.АдресДоставки, КомментарийАдресаДоставки);
КонецПроцедуры
```

7. Поместить в форму документа поля ввода, связанные с реквизитами объекта `ДатаДоставки` и `Метро`.
8. В модуле объекта документа, в обработчике события `ПередЗаписью` заполнить вспомогательные реквизиты:

```
ГородДоставки = УправлениеКонтактнойИнформацией.ГородАдресаКонтактнойИнформации(АдресДоставки);
АдресДоставкиСтрокой = УправлениеКонтактнойИнформацией.ПредставлениеКонтактнойИнформации(АдресДоставки);
```

9. БСП обеспечивает программный интерфейс для оперирования данными:

- методы для разбора контактной информации из строки представления;
- методы для формирования представления контактной информации;
- методы для получения и установки значений частей контактной информации (например, комментария, страны адреса, города адреса и т. п.);
- методы для сравнения объектов контактной информации.

Пример реализации см. в демонстрационной конфигурации в реквизите `АдресДоставки` документа `_ДемоЗаказПокупателя`.

Изменение видов контактной информации

Для изменения свойства вида контактной информации необходимо получить текущие параметры вида КИ, изменить необходимое свойство и записать эти параметры. Например, для включения хранения истории изменений адреса в обработчике обновления:

```
Вид = УправлениеКонтактнойИнформацией.ПараметрыВидаКонтактнойИнформации(Справочники.ВидыКонтактнойИнформации._ДемоАдресПартнера);
Вид.ХранитьИсториюИзменений = Истина;
УправлениеКонтактнойИнформацией.УстановитьСвойстваВидаКонтактнойИнформации(Вид);
```

Следует с осторожность менять параметры видов контактной информации, особенно у статических элементов управления, т.к. эти изменения могу привести к непредвидимому поведению или ошибкам.

Отключение неиспользуемых видов контактной информации

В случае если объект – владелец контактной информации отключен по функциональной опции, имеется возможность отключить неиспользуемый вид контактной информации. Для этого можно воспользоваться свойством `Используется` вида контактной информации. При установке значения свойства `Используется` в значение `Ложь` вид контактной информации перестает отображаться в форме списка, а также в форме объекта – владельца контактной информации. Устанавливать свойство `Используется` можно как для отдельных видов контактной информации, так и для всей контактной информации объекта владельца (группа видов контактной информации). Для отключения неиспользуемого вида контактной информации необходимо:

- Написать обработчик обновления, отключающий неиспользуемые виды контактной информации при переходе на новую версию. Пример кода обработчика:

```
КонтактныеЛицПартнеров = УправлениеКонтактнойИнформацией.ПараметрыВидаКонтактнойИнформации(Справочники.ВидыКонтактнойИнформации.СправочникКонтактныеЛицаПартнеров);
КонтактныеЛицПартнеров.Используется = ПолучитьФункциональнуюОпцию("_ДемоИспользоватьКонтактныеЛицаПартнеров");
УправлениеКонтактнойИнформацией.УстановитьСвойстваВидаКонтактнойИнформации(КонтактныеЛицПартнеров);
```

- Добавить аналогичный код в событие при изменении значения функциональной опции. См. пример в событии `ПриЗаписи` в модуле менеджера значения константы `_ДемоИспользоватьКонтактныеЛицаПартнеров`.

Добавление реквизитов типа «СправочникСсылка.СтраныМира»

В общем случае при добавлении реквизита типа `СправочникСсылка.СтраныМира` можно выбирать любой элемент справочника. Но если список выбираемых стран мира должен быть регламентирован (т. е. разрешен выбор стран только из классификатора), то дополнительно следует в свойстве `ПараметрыВыбора` элемента формы (поля ввода) указать:

- `РазрешитьДанныеКлассификатора` — `Булево`, `Истина`;
- `ТолькоДанныеКлассификатора` — `Булево`, `Истина`.

Также есть возможность автоматически создавать новые элементы справочника `СтраныМира` при выборе автоподбором. Для этого необходимо в модуле формы в обработчике события `ОбработкаВыбора` вставить вызов:

```
УправлениеКонтактнойИнформациейКлиент.СтранаМираОбработкаВыбора(Элемент, ВыбранноеЗначение, СтандартнаяОбработка);
```

Пример использования см. в форме элемента справочника `_ДемоНоменклатура` демонстрационной конфигурации.

Совместное внедрение с другими подсистемами

- Внедрение подсистемы «Пользователи» без подсистемы «Контактная информация».

Использование при разработке конфигурации

При разработке новых или доработке существующих предопределенных видов контактной информации необходимо предусмотреть их обновление при помощи обработчика обновления.

Автоматическое добавление элементов в справочник «СтраныМира» из классификатора стран мира

Пользователи с соответствующими правами могут добавлять в справочник новые элементы – как вручную, так и с помощью подбора из классификатора стран мира. При этом новые элементы справочника могут быть автоматически созданы по данным классификатора в процессе выбора и непосредственно при автоподборе в поле ввода страны.

Таким образом, в справочнике `СтраныМира` содержатся только те страны, которые используются в учете.

Настройка обмена данными

Рекомендуется включить все объекты метаданных, содержащие данные, в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места.

Для планов обмена, предназначенных для синхронизации данных между различными приложениями, нужно дополнительно настроить для справочника `СтраныМира` правила сопоставления: сначала поиск по ссылке, а затем по составному ключу `Код` и `Наименование`.

Настройка обмена данными с предыдущими версиями

- При синхронизации контактной информации с конфигурациями на базе БСП редакции 2.4 и ниже необходимо в правилах конвертации настроить правила выгрузки таким образом, чтобы для объектов – владельцев контактной информации исключать выгрузку реквизита `Значение` табличной части `КонтактнаяИнформация`.
- При синхронизации контактной информации с конфигурациями на базе БСП версии 2.1.2 и ниже необходимо при выгрузке в эту конфигурацию производить конвертацию реквизита `ЗначенияПолей` табличной части `КонтактнаяИнформация` при помощи вызова:

  ```
  Значение = РаботаСАдресами.ПредыдущийФорматКонтактнойИнформацииXML(Значение, Истина)
  ```

  При использовании правил конвертации для этого необходимо вписать приведенный выше код в обработчик `ПриВыгрузке` правила конвертации свойства (ПКС) для реквизита `ЗначенияПолей` табличной части `КонтактнаяИнформация`.
- При синхронизации контактной информации с конфигурациями на базе БСП редакции 2.2 и ниже необходимо в правилах конвертации настроить правила выгрузки из БСП 2.3.1 в БСП 2.2 таким образом, чтобы для объектов – владельцев контактной информации исключать выгрузку:
  - реквизита `ВидДляСписка` табличной части `КонтактнаяИнформация`;
  - всех строк табличной части `КонтактнаяИнформация` с типом контактной информации `Skype`.
- Если табличная часть `КонтактнаяИнформация` содержит реквизит `ДействуетС`, то при синхронизации контактной информации с конфигурациями на базе БСП версии 2.3.1 и ниже для видов контактной информации с установленным свойством `ХранитьИсториюИзменений` необходимо настроить правила обмена следующим образом:
  - при выгрузке данных – если у вида контактной информации существует несколько записей с разными датами в реквизите `ДействуетС`, то в выгрузку включается только актуальная (действующая) запись. Все исторические записи игнорируются и в выгрузку не попадают;
  - при загрузке данных – все записи этого вида контактной информации сохраняются без изменений. Но в случае, когда загружаемая запись отличается от данных текущей (действующей) записи этого вида контактной информации, в табличную часть `КонтактнаяИнформация` добавляется новая строка с загружаемыми данными, а реквизиту `ДействуетС` новой строки контактной информации присваивается текущая дата сеанса, т. к. эта запись становится актуальной (действующей).

Пример использования см. в демонстрационной конфигурации, в макетах `ПравилаОбмена` и `ПравилаОбменаКорреспондента` плана обмена _`ДемоОбменСБиблиотекойСтандартныхПодсистем225`.

Контроль ведения учета

Подсистема **Контроль ведения учета** предназначена для автоматического регламентного контроля корректности данных информационной базы по произвольным прикладным правилам, которые дополняют существующие проверки при записи документов и других объектов конфигурации (например: нарушение ссылочной целостности, отрицательные остатки в регистре накопления, сбой в нумерации счетов фактур и т. д.), а также вывода выявленных проблем для различных категорий пользователей. Например, если это операция закрытия месяца, то проверку и исправление ошибок должен проводить пользователь, компетентный в вопросах закрытия месяца; если это проверка на наличие «битых» ссылок или тестирование/исправление, то такую проверку должен проводить администратор системы и т. д. Контроль выполняется в фоне, по расписанию, или может быть явно инициирован пользователем.

Настройка

Принять решение по поводу состава прикладных проверок, которые дополнительно требуется выполнять для контроля ведения учета помимо существующих проверок заполнения при записи объектов. Например, это могут быть сложные проверки, которые контролируют состояние нескольких связанных объектов и которые целесообразно выполнять в фоне (а не во время работы пользователя), или проверки, которые следует выполнять только на определенных этапах – при операции закрытия месяца, перед отправкой отчетности и т. п. По умолчанию в подсистеме поставляются системные проверки, которые автоматически контролируют корректность данных в фоне:

- **Поиск ссылок на несуществующие файлы в томах хранения**;
- **Проверка дублирования предопределенных элементов**;
- **Проверка незаполненных обязательных реквизитов**;
- **Проверка отсутствия предопределенных узлов плана обмена**;
- **Проверка отсутствующих предопределенных элементов**;
- **Проверка ссылочной целостности**;
- **Проверка циклических ссылок**.

Для добавления прикладной проверки необходимо:

- В процедуру `ПриОпределенииПроверок` общего модуля `КонтрольВеденияУчетаПереопределяемый` внести описание проверки в следующем виде:

```
Проверка = Проверки.Добавить();
Проверка.ИдентификаторГруппы          = "СистемныеПроверки";
Проверка.Наименование                 = НСтр("ru='Демо: Проверка заполнения комментария в документах ""Демо: Поступление товаров""'");
Проверка.Причины                      = НСтр("ru='Не введен комментарий в документе.'");
Проверка.Рекомендация                 = НСтр("ru='Ввести комментарий в документе.'");
Проверка.Идентификатор                = "ПроверитьКомментарийВПоступлениитоваров";
Проверка.ОбработчикПроверки           = "_ДемоСтандартныеПодсистемы.ПроверитьКомментарийВПоступлениитоваров";
Проверка.ДатаНачалаПроверки           = Дата('20140101000000');
Проверка.ЛимитПроблем                 = 3;
Проверка.ЗапрещеноИзменениеВажности   = Истина;
Проверка.КонтекстПроверокВеденияУчета = "СистемныеПроверки";
```

- В свойстве `ОбработчикПроверки` указать полный путь экспортной процедуры – обработчика проверки в формате `<СерверныйОбщийМодуль>.<ИмяПроцедуры>`. Пример реализации см. в _ `ДемоСтандартныеПодсистемы.ПроверитьКомментарийВПоступлениитоваров`.
- Подробное описание свойств проверки находится в комментарии к процедуре `ПриОпределенииПроверок` общего модуля `КонтрольВеденияУчетаПереопределяемый`.
- В общем случае формат обработчика проверки должен иметь следующий вид:

```
Процедура ПроверитьКомментарийВПоступлениитоваров(Проверка, ПараметрыПроверки) Экспорт
    Результат = ОсуществитьПроверку();

    Пока Результат.Следующий() Цикл
        Проблема = МодульКонтрольВеденияУчета.ОписаниеПроблемы(Результат.Ссылка, ПараметрыПроверки);
        Проблема.Ответственный = Результат.Ответственный;
        Проблема.УточнениеПроблемы = ?(ЗначениеЗаполнено(Результат.Комментарий),
            НСтр("ru = 'В комментарии введены пробелы или табуляции.'"),
            НСтр("ru = 'Не введен комментарий в документе.'"));

        МодульКонтрольВеденияУчета.ЗаписатьПроблему(Проблема, ПараметрыПроверки);
    КонецЦикла;
КонецПроцедуры
```

- Вместе с проверкой подсистема также позволяет реализовать алгоритм по исправлению выявленных ее проблем. Пользователи могут переходить к исправлению из отчета `РезультатыПроверкиУчета`:
  - Для открытия формы (рабочего места), в которой предусмотрена процедура исправления в описании проверки, в свойстве `ОбработчикПереходаКИсправлению` указать:

    ```
    Проверка.ОбработчикПереходаКИсправлению = "Отчет.РезультатыПроверкиУчета.Форма.ИсправлениеЦиклическихСсылок";
    ```

  - При этом в форму будут переданы параметры `ИдентификаторПроверки` и `ВидПроверки`, по которым в коде модуля формы можно определить, исправление каких именно проблем запросил пользователь.
  - Либо в свойстве `ОбработчикПереходаКИсправлению` можно указать экспортную процедуру – обработчик клиентского общего модуля:

    ```
    Проверка.ОбработчикПереходаКИсправлению = " КонтрольВеденияУчетаСлужебныйКлиент.ИсправлениеЦиклическихСсылок";
    ```

  - При этом в экспортной клиентской процедуре-обработчике должны быть определены два параметра: `ИдентификаторПроверки` и `ВидПроверки`:

    ```
    Процедура ИсправлениеЦиклическихСсылок(ИдентификаторПроверки, ВидПроверки) Экспорт
    ```

Затем необходимо определить ссылочные объекты метаданных конфигурации (справочники, документы и т. п.), в формах и списках которых требуется выводить информацию о выявленных в них проблемах. Например, это могут быть вообще все объекты конфигурации либо только те, которые наиболее критичны для целей контроля ведения учета. Для каждого такого объекта необходимо:

- В модуле формы списка в процедуру `ПриСозданииНаСервере` добавить фрагмент:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
КонтрольВеденияУчета.ПриСозданииНаСервереФормыСписка(ЭтотОбъект, "Список");
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

- У динамического списка добавить обработчик события `СписокПриПолученииДанныхНаСервере`:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
КонтрольВеденияУчета.ПриПолученииДанныхНаСервере(Настройки, Строки);
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

- У динамического списка добавить обработчик события `Выбор`:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
&НаКлиенте
Процедура СписокВыбор(Элемент, ВыбраннаяСтрока, Поле, СтандартнаяОбработка) Экспорт

    КонтрольВеденияУчетаКлиент.ОткрытьОтчетПоПроблемамИзСписка(ЭтотОбъект, "Список", Поле, СтандартнаяОбработка);

КонецПроцедуры
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

Если обработчик события уже существует, то добавить в него вызов процедуры:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
КонтрольВеденияУчетаКлиент.ОткрытьОтчетПоПроблемамИзСписка(ЭтотОбъект, "Список", Поле, СтандартнаяОбработка);
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

- В модуле формы объекта в процедуру `ПриЧтенииНаСервере` необходимо добавить фрагмент:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
КонтрольВеденияУчета.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

- В модуле формы объекта в процедуру `ПослеЗаписиНаСервере` необходимо добавить фрагмент:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
КонтрольВеденияУчета.ПослеЗаписиНаСервере(ТекущийОбъект);
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

- В область `СлужебныеПроцедурыИФункции` модуля формы объекта поместить следующий код:

```
// СтандартныеПодсистемы.КонтрольВеденияУчета
&НаКлиенте
Процедура Подключаемый_ОткрытьОтчетПоПроблемам(ЭлементИлиКоманда, НавигационнаяСсылка, СтандартнаяОбработка)
    КонтрольВеденияУчетаКлиент.ОткрытьОтчетПоПроблемамОбъекта(ЭтотОбъект, Объект.Ссылка, СтандартнаяОбработка);
КонецПроцедуры
// Конец СтандартныеПодсистемы.КонтрольВеденияУчета
```

По умолчанию информация о выявленных проблемах выводится в списках в виде колонки с восклицательным знаком, а также в верхней части форм самих объектов. Однако это можно переопределить с помощью общего модуля `КонтрольВеденияУчетаПереопределяемый`.

Если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить команду открытия справочника `ПравилаПроверкиУчета`.

Если в конфигурации не используется подсистема **Текущие дела**, то рекомендуется уведомлять пользователей о выявленных проблемах на начальной странице другими средствами, воспользовавшись программным интерфейсом `ОткрытьОтчетПоПроблемам`, `ОткрытьОтчетПоПроблемамОбъекта` или `ОткрытьОтчетПоПроблемамИзСписка` общего модуля `КонтрольВеденияУчетаКлиент`.

Особые случаи внедрения

При внедрении подсистемы может возникнуть необходимость ограничить доступ на уровне записей к сведениям о выявленных проблемах. Например, если в конфигурации предусмотрена настройка доступа к данным в разрезе организаций и проектов, то и доступ к сведениям о выявленных проблемах также должен быть ограничен соответственным образом. Для этого необходимо:

- добавить в регистр `РезультатыПроверкиУчета` реквизиты, по которым будет осуществляться доступ, и вписать в роль `ЧтениеРезультатовПроверкиУчета` в право `Чтение` требуемые ограничения доступа к данным;
- заполнять эти реквизиты при записи проблем из кода прикладных проверок. Например, переносить их непосредственно из проблемного объекта или по более сложным правилам. При этом для поставляемых проверок следует воспользоваться процедурой `ПередЗаписьюПроблемы` общего модуля `КонтрольВеденияУчетаПереопределяемый`, в которой разместить код по заполнению следующего вида:

```
Если Реквизиты.Найти("Организация") <> Неопределено Тогда
    Проблема.Вставить("_ДемоОрганизация", ОбщегоНазначения.ЗначениеРеквизитаОбъекта(ССылкаНаОбъект, "Организация"));
КонецЕсли;
```

При использовании подсистемы может возникнуть необходимость группировать проверки по какому-либо прикладному признаку. Например, нужно выделить все проверки, которые относятся к операции расчета заработной платы. Для этого необходимо добавить в определяемый тип `КонтекстПроверокВеденияУчета` интересующий тип метаданных, после чего в процедуре `ПриОпределенииПроверок` общего модуля `КонтрольВеденияУчетаПереопределяемый` присвоить свойству `КонтекстПроверокВеденияУчета` нужное значение.

Кроме того, при необходимости этот признак (контекст проверки) может быть дополнительно уточнен еще одним значением. Например, уточнение может быть в виде вопроса: «Когда выполнять проверку?» – с возможными вариантами ответа: «До расчета заработной платы», «Во время расчета заработной платы» или «После расчета заработной платы». При необходимости такого уточнения нужно добавить в определяемый тип `УточнениеКонтекстаПроверокВеденияУчета` интересующий тип метаданных и задать одноименное свойство проверки по аналогии со свойством `КонтекстПроверокВеденияУчета`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Контроль ведения учета** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение использования подсистемы **Контроль ведения учета**. Изменение настроек выполняемых проверок.
2.	`ЧтениеРезультатовПроверкиУчета` Просмотр выявленных проблем, работа с отчетом **Контроль ведения учета**.

Использование при разработке конфигурации

Программный интерфейс подсистемы представлен в общих модулях `КонтрольВеденияУчета` и `КонтрольВеденияУчетаКлиент`.

Для просмотра выявленных проблем ведения учета подсистема включает в себя отчет `РезультатыПроверкиУчета`. В тех случаях, когда список проблем необходимо вывести в собственном рабочем месте (например, в форме закрытия месяца выводить проблемы, препятствующие закрытию месяца), следует воспользоваться процедурой `ВыполнитьПроверку` для программного выполнения нужной проверки и функциями `СводнаяИнформацияПоВидамПроверок` или `ПодробнаяИнформацияПоВидамПроверок` для получения результатов проверки. При необходимости выполнить группу проверок, связанных общим признаком (например, нужно выполнить все проверки, относящиеся к операции закрытия месяца), следует воспользоваться процедурой `ВыполнитьПроверкиВКонтексте`.

Общие рекомендации по разработке проверок

При разработке собственных проверок рекомендуется:

- исключать из проверки объекты, недоступные в данный момент по функциональным опциям, с помощью функции `ОбъектМетаданныхДоступенПоФункциональнымОпциям` общего модуля `ОбщегоНазначения`. Пример см. в процедуре `ПроверитьНезаполненныеОбязательныеРеквизиты` общего модуля `КонтрольВеденияУчетаСлужебный`;
- при работе в модели сервиса следует исключать из проверки объекты, недоступные роли `ПолныеПрава`, а также неразделенные данные с помощью функции `ЭтоРазделенныйОбъектМетаданных` общего модуля `РаботаВМоделиСервиса`. Кроме того, ряд проверок вообще не имеет смысла выполнять в модели сервиса – например, проверка ссылочной целостности отключена в модели сервиса, так как такая проверка должна выполняться централизованно администратором сервиса, а не администраторами областей данных.

Предварительная очистка проблем ведения учета перед запуском новой проверки

Для разработки проверок ведения учета, регистрирующих проблемы сразу нескольких видов (например, проблемы по указанному контрагенту или проблемы, относящиеся к определенному периоду), в общем модуле `КонтрольВеденияУчета` предусмотрена процедура `ОчиститьРезультатыПредыдущихПроверок`. С ее помощью следует предварительно очищать ранее зарегистрированные проблемы ведения учета в начале основного алгоритма обработчика проверки. В противном случае одна и та же проблема может быть зарегистрирована многократно при нескольких последовательных запусках проверки.

При этом для непараметрических проверок предварительная очистка результатов предыдущих запусков выполняется автоматически, поэтому в их обработчиках вызов процедуры `ОчиститьРезультатыПредыдущихПроверок` не требуется.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, не рекомендуется включать в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места, так как в каждом узле проверка ведения учета ведется независимо.

Контроль работы пользователей

Подсистема **Контроль работы пользователей** предназначена для анализа работы пользователей и диагностики работы приложения на основе данных журнала регистрации. Включает в себя варианты отчетов:

- **Анализ активности пользователя**, **Анализ активности пользователей**, **Анализ активности подразделений**;
- **Изменение учетных записей пользователей**;
- **Изменение прав ролей**, **Изменение ролей профилей доступа**, **Изменение участников групп пользователей**, **Изменение участников групп доступа**, **Изменение разрешенных значений групп доступа**;
- **Контроль журнала регистрации**;
- **Продолжительность работы регламентных заданий** (только локальный режим).

Также подсистема предоставляет интерфейс для аудита доступа к данным с помощью настройки события журнала регистрации **Доступ.Доступ** и рабочего места **Журнал доступа к данным**.

Настройка

Если в конфигурации нет подсистемы **Настройки программы**, то разместить в командном интерфейсе администратора:

- отчеты `АнализЖурналаРегистрации` и `ИзменениеУчетныхЗаписей`,
- команды `НастройкиРегистрацииСобытийДоступаКДанным` и `ЖурналДоступаКДанным` обработки `Журнал регистрации`. См. пример размещения в демонстрационной конфигурации в группе **Аудит доступа к данным** формы `НастройкиПользователейИПрав` обработки `ПанельАдминистрированияБСП`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Контроль работы пользователей** следует использовать роль, указанную ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Просмотр отчетов по журналу регистрации.

Использование при разработке конфигурации

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме следующих:

- константа `НастройкиРегистрацииСобытийДоступаКДанным`,
- константа `РегистрироватьИзмененияПравДоступа`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется также исключать следующие объекты метаданных:

- константа `НастройкиРегистрацииСобытийДоступаКДанным`,
- константа `РегистрироватьИзмененияПравДоступа`.

Машиночитаемые доверенности

Подсистема **Машиночитаемые доверенности** предоставляет программный и пользовательский интерфейс для работы с машиночитаемыми доверенностями в формате, соответствующем приказу Минцифры России от 18.08.2021 № 857 «Об утверждении единых требований к формам доверенностей, необходимых для использования квалифицированной электронной подписи».

Настройка

Необходимо принять решение по поводу объектов конфигурации ссылочного типа, которые могут выступать стороной МЧД, например: организации, физические лица, контрагенты. Указать их в определяемом типе `СторонаМЧД`. Эти объекты также должны быть указаны в определяемых типах `Организация`, `ФизическоеЛицо`, `Контрагент`.

Для удобства заполнения данных машиночитаемой доверенности в общем модуле `МашиночитаемыеДоверенностиФНСПереопределяемый` необходимо вписать код в процедуры `ПриЗаполненииРеквизитовОрганизации`, `ПриЗаполненииРеквизитовФизическогоЛица`, `ПриЗаполненииРеквизитовКонтрагента`, `ПриЗаполненииНалоговыхОргановДействия`. См. пример в демонстрационной конфигурации.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Машиночитаемые доверенности** следует использовать следующие роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Просмотр результатов проверки машиночитаемой доверенности в подписи.
2.	`ДобавлениеИзменениеМашиночитаемыхДоверенностей` Добавление машиночитаемых доверенностей, отправка в распределенный реестр, загрузка из реестра, загрузка и выгрузка в файл.
3.	`ЧтениеМашиночитаемыхДоверенностей` Выбор доверенности при подписании, просмотр доступных данных доверенности.
4.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Изменение всех доступных свойств любых машиночитаемых доверенностей.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы.

Мультиязычность

Группа подсистем **Мультиязычность** содержит базовую функциональность, необходимую для работы с данными на нескольких языках, а также ряд подсистем, которые расширяют возможности работы других подсистем с мультиязычными данными (например, **Печать**). При встраивании подсистемы **Мультиязычность** ее подчиненные подсистемы необходимо встраивать только совместно с соответствующими подсистемами группы **Стандартные подсистемы**, за исключением подсистемы **Перевод текста**, которая является самостоятельной и встраивается при необходимости.

При разработке конфигураций, рассчитанных на работу пользователей на нескольких языках, может возникнуть необходимость вводить и хранить значения строковых реквизитов на этих языках (далее: мультиязычные данные). В целях оптимальной производительности для мультиязычных данных доступны максимум три языка ввода данных: основной язык (может отличаться от основного языка конфигурации) и два дополнительных языка. При этом их будет предложено выбрать администратору при первом запуске информационной базы, кроме того, языки можно перевыбрать позднее из раздела **Администрирование**.

> Вне зависимости от количества языков интерфейса, подсистема **Печать** из группы подсистем **Мультиязычность** позволяет формировать печатные формы документов на неограниченном количестве языков. Инструкция по внедрению, указанная ниже, описывает предоставление возможности работы в приложении на нескольких языках интерфейса. Инструкцию по включению мультиязычности для реквизитов, используемых для печати см. в разделе Вывод печатной формы на разных языках.

Для ввода и хранения значений строковых реквизитов на разных языках необходимо:

1. Определить состав строковых реквизитов объектов ссылочного типа (справочники, документы и т.п.) и регистров сведений, для которых пользователь должен иметь возможность вводить мультиязычные данные, и создать для них реквизиты дополнительных языков. Реквизиты дополнительных языков следует называть с суффиксами `Язык1` и `Язык2`. Например, для строкового реквизита `СодержаниеЗаказа` следует создать строковые реквизиты `СодержаниеЗаказаЯзык1` и `СодержаниеЗаказаЯзык2`.

2. Для распространенных реквизитов `Наименование` и `Комментарий` в библиотеке уже предусмотрены соответствующие общие реквизиты, поэтому достаточно включить в них объекты с этими реквизитами. В остальных случаях одинаковые реквизиты дополнительных языков разных объектов рекомендуется создавать в виде общих реквизитов, в состав которых включать относящиеся к ним объекты.

3. Включить добавленные реквизиты в состав функциональных опций `ИспользоватьДополнительныйЯзык1` и `ИспользоватьДополнительныйЯзык2`.

4. Затем для каждого объекта, допускающего ввод мультиязычных данных, в форме списка в обработчике события `ПриСозданииНаСервере` вставить вызов процедуры `МультиязычностьСервер.ПриСозданииНаСервере`:

   ```
   &НаСервере
   Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
     МультиязычностьСервер.ПриСозданииНаСервере(ЭтотОбъект);
   КонецПроцедуры
   ```

   Для сложных запросов в списках, когда автоматическая адаптация запроса не применима, необходимо самостоятельно реализовать логику изменения запроса для вывода мультиязычных данных на текущем языке пользователя.

5. В обработчики событий формы элемента (документа, записи) `ПриСозданииНаСервере`, `ПриЧтенииНаСервере`, `ПередЗаписьюНаСервере` и `ПослеЗаписиНаСервере` вставить вызовы одноименных процедур их общего модуля `МультиязычностьСервер`:

   ```
   #Область ОбработчикиСобытийФормы
   &НаСервере
   Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
       МультиязычностьСервер.ПриСозданииНаСервере(ЭтотОбъект, Объект);
   КонецПроцедуры
   &НаСервере
   Процедура ПриЧтенииНаСервере(ТекущийОбъект)
       МультиязычностьСервер.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
   КонецПроцедуры
   &НаСервере
   Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
       МультиязычностьСервер.ПередЗаписьюНаСервере(ТекущийОбъект);
   КонецПроцедуры
   &НаСервере
   Процедура ПослеЗаписиНаСервере(ТекущийОбъект, ПараметрыЗаписи)
       МультиязычностьСервер.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
   КонецПроцедуры
   #КонецОбласти
   ```

   > Для регистра сведений здесь и далее переменную `Объект` следует заменить на `Запись`.

   В область `ОбработчикиСобытийЭлементовШапкиФормы` модуля формы элемента (документа) поместить следующий код:

   ```
   #Область ОбработчикиСобытийЭлементовШапкиФормы
   &НаКлиенте
   Процедура Подключаемый_Открытие(Элемент, СтандартнаяОбработка)
       МультиязычностьКлиент.ПриОткрытии(ЭтотОбъект, Объект, Элемент, СтандартнаяОбработка);
   КонецПроцедуры
   #КонецОбласти
   ```

6. В модуль объекта вставить процедуру `ПриЧтенииПредставленийНаСервере`:

   ```
   #Область ОбработчикиСобытий
   Процедура ПриЧтенииПредставленийНаСервере() Экспорт
       МультиязычностьСервер.ПриЧтенииПредставленийНаСервере(ЭтотОбъект);
   КонецПроцедуры
   #КонецОбласти
   ```

7. Для объекта ссылочного типа (справочники, документы и т.п.) в модуль менеджера объекта добавить обработчики `ОбработкаПолученияПолейПредставления`, `ОбработкаПолученияПредставления` и `ОбработкаПолученияДанныхВыбора`:

```
#Область ОбработчикиСобытий
Процедура ОбработкаПолученияПолейПредставления(Поля, СтандартнаяОбработка)
    МультиязычностьКлиентСервер.ОбработкаПолученияПолейПредставления(Поля, СтандартнаяОбработка);
КонецПроцедуры
Процедура ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка)
    МультиязычностьКлиентСервер.ОбработкаПолученияПредставления(Данные, Представление, СтандартнаяОбработка);
КонецПроцедуры
Процедура ОбработкаПолученияДанныхВыбора(ДанныеВыбора, Параметры, СтандартнаяОбработка)
    МультиязычностьСервер.ОбработкаПолученияДанныхВыбора(ДанныеВыбора, Параметры, СтандартнаяОбработка, <Метаданные.ПланыВидовХарактеристик
КонецПроцедуры
#КонецОбласти
```

Четвертым параметром в вызове процедуры `МультиязычностьСервер.ОбработкаПолученияДанныхВыбора` следует передавать объект метаданных.

Пример реализации см. в демонстрационной конфигурации в справочнике `РолиИсполнителей` и регистре сведений `_ДемоЗаведующиеМестамиХранения`.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы.

Перевод текста

В состав группы подсистем **Мультиязычность** входит подсистема **Перевод текста**, которая предоставляет возможность автоматического перевода текста на другой язык с помощью внешних сервисов Yandex.Cloud и Google Cloud. Кроме того, она дополняет подсистему **Печать** для вывода печатных форм на разных языках, а также позволяет автоматически переводить представления мультиязычных реквизитов.

Программный интерфейс подсистемы представлен в общем модуле `ПереводТекстаНаДругиеЯзыки`.

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Настройка сервиса перевода текста, перевод текста.
2.	`БазовыеПрава` (из подсистемы **Базовая функциональность**) Перевод текста.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между приложениями, рекомендуется включать все объекты метаданных подсистемы, кроме:

- константа `ИспользоватьСервисПереводаТекста`,
- константа `СервисПереводаТекста`.
- регистр сведений `КэшПереводов`.

Это вызвано тем, что в каждом узле или приложении настройка сервиса перевода выполняется независимо.

Напоминания пользователя

Подсистема **Напоминания пользователя** предназначена для установки персональных напоминаний по поводу какого-либо объекта системы и оповещения пользователя в назначенное время.

Настройка

Для использования подсистемы необходимо разместить в командном интерфейсе пользователя форму `МоиНапоминания` регистра сведений `НапоминанияПользователя`, а также выполнить следующие действия:

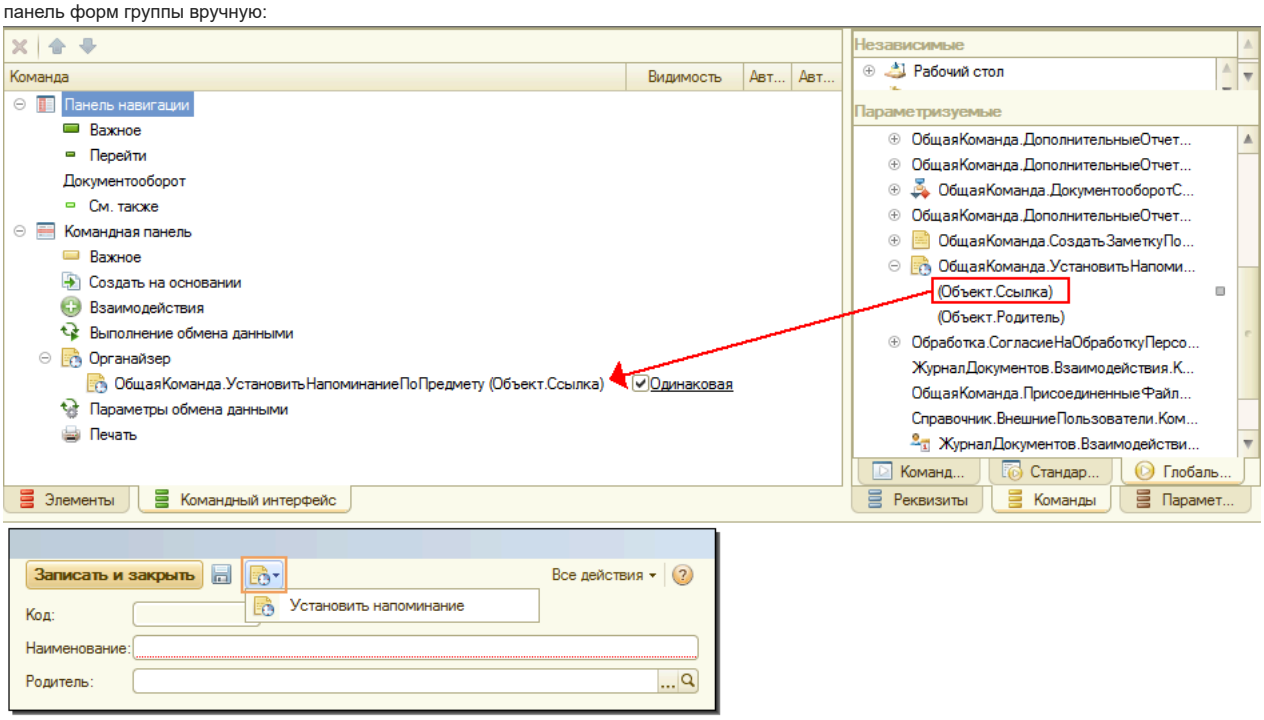
1. Определить состав объектов ссылочного типа, по поводу которых пользователь должен иметь возможность вводить напоминания. Как правило, это большинство объектов конфигурации, с которыми работает пользователь.
2. В определяемых типах `ПредметНапоминания` и `ПредметНапоминанияОбъект` отметить выбранные типы объектов в свойстве `Тип`.
3. Если в конфигурацию включена подсистема `ПодключаемыеКоманды`, команды напоминаний выводятся с ее помощью. В этом случае в формах, в которых должны отображаться кнопки напоминаний, следует внедрить механизм подключаемых команд.

Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд напоминаний по шаблону:

- Имя: `ПодменюОрганайзер`.
- Заголовок: **Органайзер**.
- Вид: `Подменю`.
- Отображение: `Картинка`.

- Картинка: `Органайзер` (картинка из конфигурации).

> **Примечание**
>
> В случае отсутствия подсистемы `ПодключаемыеКоманды` в конфигурации, команды напоминаний выводятся с помощью общих команд. По умолчанию общая команда `Напомнить` не видна в формах групп объектов. При необходимости эту команду нужно поместить на командную панель форм группы вручную:



Определение состава реквизитов объекта для относительной настройки срока напоминания

По умолчанию пользователю предлагается задать срок напоминания относительно любого реквизита типа `Дата`. В списке доступных для выбора реквизитов доступны только реквизиты, значения дат которых находятся в будущем. Этот список можно переопределить – например, убрать из него служебные реквизиты. Для этого необходимо вписать свою реализацию в процедуру `ПриЗаполненииСпискаРеквизитовИсточникаСДатамиДляНапоминания` модуля `НапоминанияПользователяПереопределяемый`.

Размещение элементов настройки напоминания на форме объекта

В некоторых сценариях требуется размещение настройки напоминания о дате объекта непосредственно в форме объекта (например, запись календаря). Для упрощения реализации таких сценариев в подсистеме предусмотрено автоматическое размещение элементов настройки напоминания на форме объекта.

Для размещения элементов напоминания на форме необходимо в модуле формы сделать вставки кода по следующему образцу:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    ПараметрыРазмещения = НапоминанияПользователя.ПараметрыРазмещения();
    ПараметрыРазмещения.Группа = Элементы.ГруппаДатаОптатыСНапоминанием;
    ПараметрыРазмещения.ИмяРеквизитаСДатойСобытия = "ДатаОплаты";
    НапоминанияПользователя.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыРазмещения);
КонецПроцедуры

&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)
    НапоминанияПользователя.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
КонецПроцедуры

&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
    ТекстНапоминания = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
        НСтр("ru = 'Проверить оплату по счету %1'"), ТекущийОбъект.Номер);
    НапоминанияПользователя.ПриЗаписиНаСервере(ЭтотОбъект, Отказ, ТекущийОбъект, ПараметрыЗаписи, ТекстНапоминания);
КонецПроцедуры

&НаКлиенте
Процедура ОбработкаОповещения(ИмяСобытия, Параметр, Источник)
    НапоминанияПользователяКлиент.ОбработкаОповещения(ЭтотОбъект, ИмяСобытия, Параметр, Источник);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ПриИзмененииНастройкиНапоминания(Элемент)
    НапоминанияПользователяКлиент.ПриИзмененииНастройкиНапоминания(Элемент, ЭтотОбъект);
КонецПроцедуры
```

- См. пример в форме `ФормаДокумента` документа `_ДемоСчетНаОплатуПокупателю` в демонстрационной конфигурации.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Напоминания пользователя» без подсистемы «Управление доступом».

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Создание, редактирование напоминаний пользователей, удаление напоминаний.
2.	`ДобавлениеИзменениеНапоминаний`. Создание, просмотр и удаление своих напоминаний.

Использование при разработке конфигурации

Программный интерфейс для создания напоминаний

Для неинтерактивного создания напоминаний предусмотрен программный интерфейс: процедуры `НапомнитьВУказанноеВремя` и `НапомнитьДоВремениПредмета` общего модуля `НапоминанияПользователяКлиент`. Пример его использования можно посмотреть в демонстрационной конфигурации:

- в разделе **Бизнес процессы и задачи** – **Исполнение задач** – **Мои задачи** – в любой задаче на командной панели – **Органайзер** – **Демо: Напомнить за 5 минут**;
- в разделе **Органайзер** – **Заметки пользователя** – **Демо: Контрагенты** – в карточке контрагента на командной панели – **Органайзер** – **Демо: Напомнить через 10 минут**.

Общая команда `_ДемоНапомнитьЗа5Минут`, справочник `_ДемоКонтрагенты`, команда `_ДемоНапомнитьЧерез10Минут`.

Настройка обмена данными

Все объекты метаданных подсистемы не должны включаться в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места. Работа с подсистемой в каждом узле ведется независимо.

Настройка порядка элементов

Подсистема **Настройка порядка элементов** предназначена для изменения порядка следования элементов справочника, плана видов характеристик или плана видов расчета в динамических списках. Если справочник подчиненный, то нумерация элементов выполняется в пределах владельца. Если справочник или ПВХ иерархический, то нумерация элементов выполняется в пределах родителя.

Подсистема накладывает некоторые ограничения на настройку динамического списка: упорядочивание списка должно быть по специальному реквизиту порядка, а также нежелательно использование группировок. В случаях, когда настройки списка не позволяют выполнить перемещение элемента, будет выдано соответствующее предупреждение.

Кроме того, не рекомендуется использовать подсистему в таблицах с большим количеством элементов (больше 100), так как ручное упорядочивание такого количества элементов нецелесообразно (с точки зрения прикладной задачи) и сильно замедлит пересчет порядковых номеров в списке.

Настройка

Определить состав справочников и планов видов характеристик, для которых требуется настраивать порядок элементов, указав их в составе определяемого типа `ОбъектСНастраиваемымПорядком`.

Для каждого объекта, для которого предусмотрена настройка порядка элементов:

1. Добавить реквизит `РеквизитДопУпорядочивания`.

Свойство	Значение
Тип значения	`Число`, длина – 5, точность – 0 (длина может быть изменена)
Индексировать	**Индексировать с доп. упорядочиванием**
Синоним	**Порядок**

1. Форму списка подключить к подсистеме Подключаемые команды.
2. В форме списка для динамического списка установить сортировку по полю `Порядок`.

Если у объекта есть предопределенные элементы, то в процедуре обновления информационной базы необходимо перезаписать эти элементы в том порядке, в каком они должны отображаться в списке.

> **Важно!**
>
> В списках, у которых есть отборы, а также для объектов с включенным ограничением прав на уровне записей подсистема может использоваться в ограниченном количестве случаев. Например, если каждый пользователь списка видит только свои элементы, то перемещение любого элемента будет всегда корректным. В остальных случаях возможна ситуация, когда один пользователь двигает элемент в своей области видимости на одну позицию, а для другого пользователя это «движение» выглядит как «прыжок» элемента сразу на несколько позиций.

Настройка прав доступа пользователей

Настройка прав доступа пользователей к данным подсистемы **Настройка порядка элементов** не требуется.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Настройки программы

Подсистема **Настройки программы** предоставляет панели настроек для всех подсистем библиотеки, размещаемые в разделе командного интерфейса **Администрирование**. Панели настроек представлены формами обработки `ПанельАдминистрированияБСП`.

Настройка

Расположить подсистему `Администрирование` последней (крайней права) в списке разделов командного интерфейса (свойства конфигурации – **Командный интерфейс**).

Если состав раздела **Администрирование** для объектов БСП не был загружен из поставки, то в разделе **Администрирование** следует разместить все панели администрирования (формы обработки `ПанельАдминистрированияБСП`), а также частотные команды, которые должны быть на виду (например, удаление помеченных объектов). Пример размещения см. в демонстрационной базе.

Если в конфигурацию не встроена ни одна подсистема, входящая в форму (в этом случае форма панели администрирования открывается пустой), необходимо удалить эту форму из конфигурации. Например, если не внедрена подсистема `РаботаСФайлами`, то необходимо удалить форму `НастройкиРаботыСФайлами`.

Необходимо принять решение по поводу состава настроек конфигурации и места их размещения на панелях. При выборе панели настроек нужно руководствоваться следующими соображениями:

- форма `Обслуживание` предназначена для операций, которые периодически должен выполнять администратор приложения:
  - мониторинг состояния системы;
  - резервное копирование и восстановление;
  - оптимизация быстродействия;
  - другие операции.
- форма `ОбщиеНастройки` предназначена для изменения общих для приложения параметров, например, заголовка окна приложения, полнотекстового поиска и т. п.;
- форма `НастройкиПользователейИПрав` предназначена для администрирования пользователей:

    - ведение списка пользователей;
    - настройка подсистемы прав доступа;
    - управление настройками пользователей;
    - установка дат запрета изменения.
- форма `ИнтернетПоддержкаИСервисы` предназначена для сервисов, доступных по сети Интернет:
    - загрузка классификаторов;
    - онлайн-поддержка;
    - проверка контрагентов.
- форма `НастройкиРаботыСФайлами` предназначена для настройки параметров подсистем, работающих с файлами;
- форма `НастройкиСинхронизацииДанных` предназначена для настройки синхронизации данных с другими приложениями и организации совместной работы в распределенной информационной базе;
- форма `Органайзер` предназначена для настройки электронной почты, заметок, напоминаний и бизнес-процессов;
- форма `ПечатныеФормыОтчетыИОбработки` предназначена для настройки печатных форм, вариантов отчетов, рассылок отчетов и дополнительных отчетов и обработок.

На панелях настроек могут быть размещены поля редактирования констант, гиперссылки для перехода к связанным формам настроек, поясняющие надписи и другие элементы управления.

Размещение констант

В формах настроек предусмотрен код, позволяющий упростить процесс размещения констант. Для размещения константы необходимо:

1. Перетащить константу из реквизита формы `НаборКонстант` в элементы формы. Полю ввода должно быть назначено то же имя, которое имеет константа.
2. Определить обработчик поля ввода `ПриИзменении`, разместив в нем вызов процедуры `Подключаемый_ПриИзмененииРеквизита`, например:

```
&НаКлиенте
Процедура ИмяКонстантыПриИзменении(Элемент)
    Подключаемый_ПриИзмененииРеквизита(Элемент);
КонецПроцедуры
```

Размещение гиперссылок

Для размещения гиперссылки необходимо:

1. Включить глобальную команду или объект с командами в состав раздела **Администрирование**.
2. Выключить видимость команды (объекта) в этом разделе, для того чтобы системный администратор имел возможность включить ее обратно для часто используемых команд.
3. В форме настройки перетащить команду в элементы формы. Установить вид поля `Гиперссылка`.

Размещение поясняющих надписей

Для всех полей ввода и гиперссылок рекомендуется добавлять расширенные подсказки, поясняющие назначение элемента управления. При добавлении расширенной подсказки необходимо установить у владельца свойство **Отображение подсказки** в значение **Отображать снизу**, а также отключить у расширенной подсказки свойство `АвтомаксимальнаяШирина`.

Поддержка различных режимов работы конфигурации

Если конфигурация рассчитана на различные режимы работы, следует управлять видимостью элементов управления на формах настроек. Например, гиперссылка для перехода к форме **Параметры администрирования ИБ** должна быть видна только в клиент-серверном режиме работы.

Элементы формы, связанные с функциональными опциями, скрываются автоматически. Если элементы формы не скрываются автоматически (например, если это поясняющий текст, который не может быть включен в состав функциональной опции), то в обработчике `ПриСозданииНаСервере` следует написать код, который будет скрывать всю группу.

Для определения режима работы конфигурации в поставляемых формах настроек для реквизита формы `РежимРаботы` можно использовать свойства: `МодельСервиса`, `Локальный`, `Файловый`, `КлиентСерверный`, `ЛокальныйФайловый`, `ЛокальныйКлиентСерверный` (подробнее см. описание процедуры `ПанельНастроекЗаполнитьРежимРаботы` общего модуля `НастройкиПрограммыСервер`).

Например:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    // Настройки видимости при запуске
    Элементы.ГруппаПараметрыАдминистрированияИБ.Видимость = РежимРаботы.КлиентСерверный;

КонецПроцедуры
```

Взаимосвязанные элементы управления

Некоторые элементы могут быть связаны друг с другом, например:

- гиперссылка для перехода к списку **Настройки версионирования объектов** должна быть доступна, только когда установлен флажок **Версионирование объектов**;
- флажок **Подчиненные Бизнес-процессы** должен быть доступен только тогда, когда установлен флажок **Бизнес-процессы и задачи**.

В таких ситуациях подчиненные элементы следует делать недоступными (но оставлять видимыми) в зависимости от главного элемента. Для этого нужно поместить обслуживающий код в процедуре `УстановитьДоступность`. Например:

```
&НаСервере
Процедура УстановитьДоступность(РеквизитПутьКДанным = "")

    Если РеквизитПутьКДанным = "НаборКонстант.ИспользоватьБизнесПроцессыИЗадачи" ИЛИ РеквизитПутьКДанным = "" Тогда

        Элементы.ОткрытьРолиИИсполнителиБизнесПроцессов.Доступность = НаборКонстант.ИспользоватьБизнесПроцессыИЗадачи;

    КонецЕсли;

КонецПроцедуры
```

Ролевая видимость

В некоторых сценариях с одной формой настроек могут работать пользователи с различными правами доступа. Например, в модели сервиса настройку приложения могут выполнять и администратор абонента, и администратор сервиса.

В таких случаях формы настроек должны «разрезаться» по правам доступа таким образом, чтобы пользователю (администратору) были видны только те настройки, к которым он имеет доступ.

Для этого необходимо добавить проверку ролей перед вызовом кода, который изменяет свойства элементов из скрываемой группы в процедурах `ПриСозданииНаСервере` или `Доступность` в процедуре `УстановитьДоступность`. Например, для проверки того, что с формой работает администратор сервиса:

```
&НаСервере
Процедура УстановитьДоступность(РеквизитПутьКДанным = "")
    Если РежимРаботы.ЭтоАдминистраторСистемы Тогда
        <Обслуживающий код>
    КонецЕсли;
КонецПроцедуры
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Доступ ко всем прикладным настройкам приложения и настройкам, специфичным только в локальном режиме работы.
2.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Доступ ко всем общим настройкам приложения при работе в модели сервиса, а также ко всем административным настройкам в локальном режиме.

Особые случаи внедрения подсистемы

Совместное внедрение с другими подсистемами

- Внедрение подсистемы «Настройки программы» без подсистемы «Работа с файлами».

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Обмен данными

Подсистема **Обмен данными** объединяет функциональность, связанную с обменом информацией между различными информационными базами:

- распределенные информационные базы (РИБ);
- обмен данными через универсальный формат;
- обмен данными по правилам обмена (правила обмена создаются при помощи конфигурации **Конвертация данных**, редакция 2.1);
- обмен данными без правил обмена.

Поддерживается обмен данными между конфигурациями, работающими в модели сервиса, а также между конфигурациями, работающими в модели сервиса и в локальном режиме. Обмен данными может быть односторонним или двусторонним и выполняться автоматически (по некоторому расписанию) или вручную по требованию пользователя. Для транспорта сообщений обмена могут быть использованы различные каналы связи: сетевой каталог, электронная почта, FTP, обмен через Интернет (веб-сервис). При использовании правил обмена, а также при обмене через универсальный формат доступно подключение к информационной базе-корреспонденту через внешнее соединение.

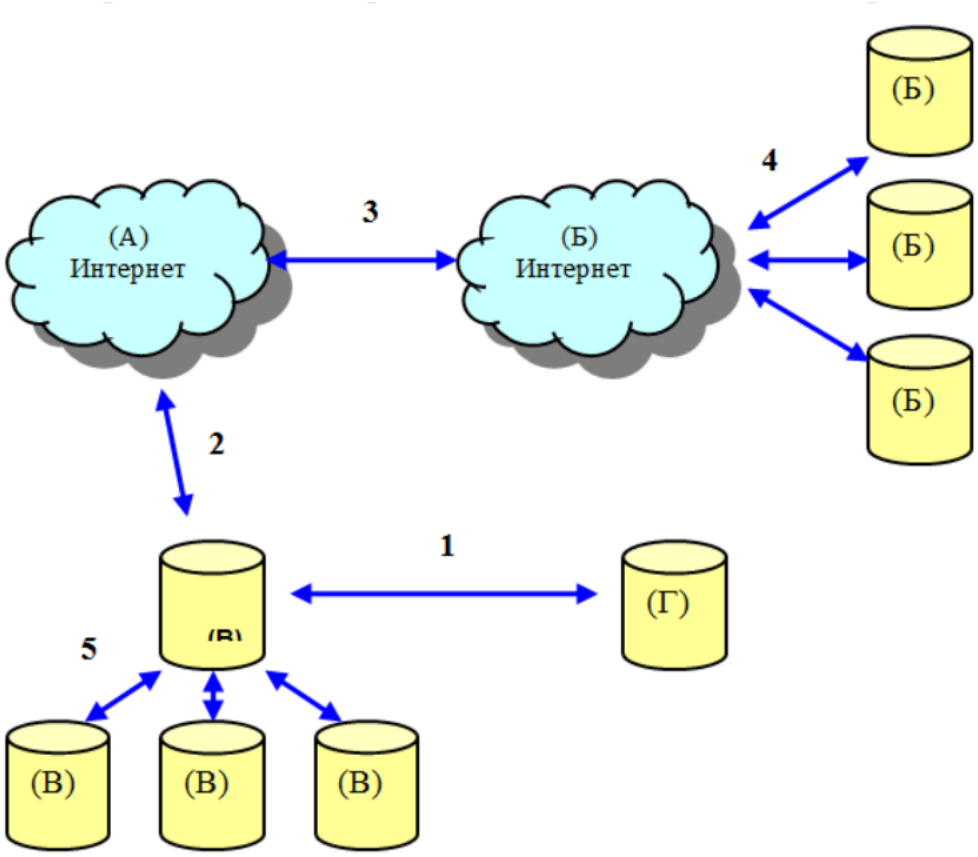
Подсистема **Обмен данными** не является самостоятельно функционирующей подсистемой. Для ее использования необходимо разработать в конфигурации прикладные планы обмена согласно предлагаемой методике и провести дополнительную настройку библиотечных объектов.

Если требуется выполнять обмен данными в модели сервиса, то дополнительно необходимо встроить подсистему **Обмен данными в модели сервиса**.

Подсистема **Обмен данными** совместно с подсистемой **Обмен данными в модели сервиса** позволяет поддержать пять сценариев обмена данными:

1. Обмен данными между различающимися конфигурациями, работающими в локальном режиме.
2. Обмен данными между различающимися конфигурациями, одна из которых работает в модели сервиса, а вторая – в локальном режиме.
3. Обмен данными между различающимися конфигурациями, работающими в модели сервиса.
4. Автономная работа в модели сервиса.
5. Обмен в распределенной информационной базе.

Поддерживаемые сценарии обмена данными наглядно представлены на рисунке ниже.



Поддерживаемые сценарии обмена данными

Подсистема предоставляет в распоряжение разработчика четыре вида обмена данными:

- обмен данными в распределенной информационной базе (**РИБ**) – используется стандартный механизм платформы для организации обмена в распределенной информационной базе;
- обмен данными через универсальный формат (**ОУФ**) – позволяет организовать обмен между различающимися конфигурациями с использованием модуля (**менеджер обмена**), в котором прописана логика приведения данных конфигурации к структуре, описанной в формате (**конфигурация** – **формат**), а также обратный процесс преобразования (**формат** – **конфигурация**);
- универсальный обмен данными по правилам (**УОП**) – позволяет организовать обмен между различающимися конфигурациями с использованием правил обмена данными, в которых прописана логика преобразования данных одной конфигурации в данные другой конфигурации;
- универсальный обмен данными без использования правил обмена (**УО**) – позволяет организовать обмен между различающимися конфигурациями. При выборе данного вида обмена следует учитывать основное требование: необходимо, чтобы объекты, которые участвуют в обмене, в обеих базах имели идентичную структуру метаданных.

Настройка

Принятие решения о виде обмена данными

Перед внедрением подсистемы следует принять решение о виде обмена данными, который необходимо реализовать. Возможности и особенности поддерживаемых видов обмена приведены в таблице ниже.

Характеристика	ОУФ	УОП	УО	РИБ
Режим использования				
Работа в локальном режиме	+	+	+	+
Работа в модели сервиса	+	+	+	+
Поддерживаемые каналы связи				
Через сетевой или локальный каталог	+	+	+	+
Через **ftp-сервер**	+	+	+	+
Через электронную почту (**e-mail**)	+	+	+	+
Обмен через Интернет (**web-сервис, активное подключение**)	+	+	+	+ [1]
Обмен через Интернет (**web-сервис, пассивное подключение**)	+	–	–	–
Обмен через прямое подключение к информационной базе-корреспонденту (**COM-соединение**)	+	+	–	–
Особенности взаимодействия конфигураций				
Существует разделение баз на главную и подчиненную	–	–	–	+
Конфигурации обменивающихся информационных баз идентичны как по структуре метаданных, так и по прикладной бизнес-логике	–	–	– [2]	+
Требуется разработка логики преобразования данных	+	+	–	–
Логика конвертации данных на стороне конфигурации № 1 не зависит от внутреннего устройства и бизнес-логики конфигурации № 2	+	–	–	–
Прочие возможности				
Настройка ограничения и направления миграции данных по узлам	+	+	+	+
Предварительный просмотр загружаемых данных и сопоставление объектов информационных баз	+	+	–	–

> **Примечание**
>
> [1] Используется для организации автономной работы в модели сервиса.
>
> [2] Конфигурации обменивающихся информационных баз должны быть идентичны только в тех метаданных, которые участвуют в обмене.

В зависимости от вида обмена технологии внедрения и настройки подсистемы в конфигурации различаются.

Настройка для работы в локальном режиме

Для использования подсистемы в конфигурации необходимо:

- разместить в командном интерфейсе общую команду `НастройкиСинхронизацииДанных`;
- в форме редактирования настроек системы поместить поле, связанное с константами `ИспользоватьСинхронизациюДанных`, `ПрефиксУзлаРаспределеннойИнформационнойБазы`;
- если предполагается использование обмена через Интернет, выполнить публикацию веб-сервисов:
  - `Exchange`,
  - `Exchange_2_0_1_6`,
  - `Exchange_3_0_1_1`.
- если предполагается использовать ОУФ через Интернет в пассивном режиме, выполнить публикацию веб-сервисов:
  - `EnterpriseDataExchange_1_0_1_1` – для обмена через план обмена с поддержкой квитирования,
  - `EnterpriseDataUpload_1_0_1_1` – для обмена без использования плана обмена и без квитирования.

Пример размещения объектов подсистемы в командном интерфейсе можно посмотреть в демонстрационной конфигурации.

В общем модуле `ОбменДаннымиПереопределяемый` необходимо задать возвращаемое значение функции `ПриОпределенииПрефиксаИнформационнойБазыПоУмолчанию`. В качестве возвращаемого значения функции следует указать строку префикса – **Строка, 2**. Длина префикса не должна превышать двух символов. Значение функции будет использоваться для задания префикса информационной базы по умолчанию в помощнике настройки обмена данными.

В общем модуле `ОбменДаннымиПереопределяемый` в процедуре `ПриПолученииДоступныхВерсийФормата` необходимо указать перечень поддерживаемых версий формата `EnterpriseData` и соответствующих им модулей-обработчиков, для чего заполнить соответствие `ВерсииФормата`, указав в качестве ключа номер версии формата, а в качестве значения – общий модуль, содержащий правила конвертации.

Настройка для работы в модели сервиса

> **Важно!**
>
> Работа подсистемы в модели сервиса возможна только при наличии подсистем `ОбменСообщениями` и `ОчередьЗаданий`, входящих в состав библиотеки технологии сервиса (БТС).

- Разместить в командном интерфейсе общую команду `АвтономнаяРабота`.
- Разместить в командном интерфейсе общую команду `АвтономнаяРаботаВМоделиСервиса`.
- Разместить в командном интерфейсе общую команду `НастройкиСинхронизацииДанных`.
- Выполнить внутреннюю неразделенную публикацию веб-сервисов:
  - `RemoteAdministrationOfExchange`,
  - `RemoteAdministrationOfExchange_2_0_1_6`,
  - `RemoteAdministrationOfExchange_2_1_6_1`,
  - `RemoteAdministrationOfExchange_2_4_5_1`.
- Выполнить внутреннюю неразделенную и внешнюю разделенную публикацию веб-сервисов:
  - `Exchange`,
  - `Exchange_2_0_1_6`,
  - `Exchange_3_0_1_1`,
  - `InterfaceVersion`.
- Если предполагается использовать ОУФ через Интернет в пассивном режиме, выполнить публикацию веб-сервисов:
  - `EnterpriseDataExchange_1_0_1_1` – для обмена через план обмена с поддержкой квитирования,
  - `EnterpriseDataUpload_1_0_1_1` – для обмена без использования плана обмена и без квитирования.

Пример файла разделенной публикации веб-сервисов `Exchange`, `Exchange_2_0_1_6 и InterfaceVersion`:

```
<?xml version="1.0" encoding="UTF-8"?>
<point xmlns="http://v8.1c.ru/8.2/virtual-resource-system"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        base="/demo"
        ib="Srvr=&quot;server1C&quot;;Ref=&quot;demo&quot;;">
    <zones>
        <zone specify="false" safe="true"/>
        <zone specify="true" safe="true"/>
    </zones>
    <ws>
        <point name="Exchange"
                alias="exchange.1cws"/>
        <point name="Exchange_2_0_1_6"
                alias="exchange_2_0_1_6.1cws"/>
        <point name="InterfaceVersion"
                alias="InterfaceVersion.1cws"/>
    </ws>
</point>
```

Пример файла неразделенной публикации веб-сервисов `RemoteAdministrationOfExchange`, `RemoteAdministrationOfExchange_2_0_1_6`, `Exchange`, `Exchange_2_0_1_6`:

```
<?xml version="1.0" encoding="UTF-8"?>
<point xmlns="http://v8.1c.ru/8.2/virtual-resource-system"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        base="/demo_ws"
        ib="Srvr=&quot;server1C&quot;;Ref=&quot;demo&quot;;"
        enable="false">
    <ws>
        <point name="Exchange"
                alias="exchange.1cws"/>
        <point name="Exchange_2_0_1_6"
                alias="exchange_2_0_1_6.1cws"/>
        <point name="RemoteAdministrationOfExchange"
                alias="RemoteAdministrationOfExchange.1cws"/>
        <point name="RemoteAdministrationOfExchange_2_0_1_6"
                alias="RemoteAdministrationOfExchange_2_0_1_6.1cws"/>
    </ws>
</point>
```

Автономная работа в модели сервиса

Одним из возможных сценариев использования подсистемы обмена данными является сценарий автономной работы в модели сервиса. Этот сценарий позволяет работать с информационной базой без постоянного подключения к Интернету и при необходимости выполнять синхронизацию данных с Интернетом.

Для организации автономной работы в конфигурации необходимо:

- создать план обмена РИБ согласно этой документации;
- план обмена должен быть разделенным (входить в состав реквизита-разделителя конфигурации `ОбластьДанныхОсновныеДанные`);
- для плана обмена должен быть установлен признак **Распределенная ИБ**;

- тип кода плана обмена: `Строка`, 36, переменная;
- в модуле менеджера плана обмена в процедуре `ПриПолученииНастроек` свойству `ПланОбменаИспользуетсяВМоделиСервиса` структуры `Настройки` следует установить значение `Истина`.

> **Важно!**
>
> В конфигурации может быть создан только один план обмена, который предназначен для обслуживания автономной работы в модели сервиса.

В состав плана обмена автономной работы не рекомендуется включать объекты метаданных классификаторов с потенциально большим объемом данных (например, адресный классификатор), т. к. обновление таких классификаторов требует много времени на синхронизацию и порождает большой трафик данных. Данные таких классификаторов рекомендуется загружать в автономное рабочее место из альтернативных источников, например с сайта «1С», и только при необходимости.

Обмен данными с автономным рабочим местом имеет следующие особенности:

- информация о пользователях и доступе к данным мигрирует только сверху вниз – из сервиса в автономное рабочее место. Все изменения, сделанные в автономном рабочем месте, игнорируются и приводятся в соответствие со значениями в сервисе;
- информация общих данных (классификаторов банков, классификаторов валют и пр.) мигрирует только сверху вниз – из сервиса в автономное рабочее место. Все изменения, сделанные в автономном рабочем месте, игнорируются и приводятся в соответствие со значениями в сервисе.

Разработка планов обмена

Для организации обмена в локальном режиме и обмена в модели сервиса используется одна технология. Поэтому разработка планов обмена и других объектов метаданных будет одинаковой для двух режимов использования. Особенности и различия будут отражены отдельно.

План обмена может содержать реквизиты и табличные части. Состав плана обмена определяет набор объектов метаданных, которые будут участвовать в обмене данными.

В общем модуле `ОбменДаннымиПереопределяемый` в тело процедуры `ПолучитьПланыОбмена` следует добавить код, относящийся к новому плану обмена:

```
Процедура ПолучитьПланыОбмена(ПланыОбменаПодсистемы) Экспорт

    ПланыОбменаПодсистемы.Добавить(Метаданные.ПланыОбмена.<ИмяПланаОбмена>);

КонецПроцедуры
```

Реквизиты и свойства плана обмена

Наименование для плана обмена необходимо назначать согласно общим правилам наименования объектов метаданных. Свойства плана обмена необходимо задать согласно приведенной ниже таблице.

Свойство	План обмена УОП и УО	План обмена ОУФ	План обмена РИБ
Имя	`Обмен<Источник><Приемник>`	Не регламентируется	Не регламентируется
Распределенная ИБ	`Ложь`	`Ложь`	`Истина`
Состав	См. описание ниже	См. описание ниже	См. описание ниже
Код (тип, длина)	`Строка`, 9, переменная	`Строка`, 36, переменная	`Строка`, 9, переменная (`Строка`, 36, переменная) [1]
Наименование	`Строка`, 150	`Строка`, 150	`Строка`, 150
Макеты	См. описание ниже	См. описание ниже	См. описание ниже

> [1] Если план обмена используется для организации автономной работы в модели сервиса.

Для плана обмена допустимо создавать произвольное количество реквизитов и табличных частей. Все реквизиты и табличные части плана обмена можно использовать в правилах регистрации объектов на узлах. Правила регистрации настраиваются при помощи конфигурации **Конвертация данных**. Правила регистрации используются как для универсальных обменов, так и для обмена в РИБ.

Для планов обмена может быть предусмотрен набор реквизитов для переключения режимов выгрузки объектов метаданных (реквизиты-переключатели). Например, для переключения режима выгрузки информации, связанной с контрагентами, создается реквизит-переключатель, который управляет режимом выгрузки сразу для нескольких объектов метаданных: справочники **Контрагенты**, **Договоры контрагентов**. Количество таких реквизитов-переключателей не ограничено. Тип реквизита-переключателя строго регламентирован – `ПеречислениеСсылка.РежимыВыгрузкиОбъектовОбмена`.

Если план обмена используется для организации обменов данными в модели сервиса и является разделенным, то для такого плана обмена следует создать реквизит `РегистрироватьИзменения`:

Имя	Тип	Индексировать
`РегистрироватьИзменения`	`Булево`	Да

Для планов обменов РИБ, которые используются в локальном режиме работы, следует установить свойство плана обмена `Включать расширение конфигурации` в значение `Истина`. При этом для планов обмена автономного рабочего места (АРМ) это свойство должно быть снято.

> **Важно!**
>
> Для разработчиков прикладных решений важно понимать, что если свойство `Включать расширение конфигурации` изменяется на Истина в существующем плане обмена РИБ, то необходимо пользователям донести следующую информацию: В работающих обменах РИБ, на момент включения свойства `Включать расширение конфигурации` в главном и периферийных узлах не должно быть расширений, меняющих структуру данных. Также следует удалить из периферийных информационных баз расширения, имена которых совпадают с расширениями в главном узле (кроме патчей). Добавлять расширения можно после обновления всех периферийных баз.

Состав плана обмена

Для планов обмена **УОП** необходимо исключить все объекты метаданных подсистем **Обмен данными** и **Обмен данными в модели сервиса**, кроме регистра сведений `СоответствияОбъектовИнформационныхБаз` (указанный регистр сведений обязателен для включения в план обмена).

> **Важно!**
>
> Все элементы состава плана обмена должны иметь признак авторегистрации `Запретить` вне зависимости от наличия правил регистрации для объектов.

Для планов обмена РИБ рекомендуется исключить все объекты метаданных подсистем **Обмен данными** и **Обмен данными в модели сервиса**, кроме константы `АдресДляВосстановленияПароляУчетнойЗаписи`. Константу `АдресДляВосстановленияПароляУчетнойЗаписи` рекомендуется включать только в состав начального образа подчиненного узла распределенной ИБ.

Для планов обмена **РИБ** с фильтрами рекомендуется исключить все объекты метаданных подсистем **Обмен данными** и **Обмен данными в модели сервиса**, кроме констант `ДанныеДляОтложенногоОбновления` и `АдресДляВосстановленияПароляУчетнойЗаписи`. Константу `АдресДляВосстановленияПароляУчетнойЗаписи` рекомендуется включать только в состав начального образа подчиненного узла распределенной ИБ.

Для остальных планов обмена рекомендуется исключить все объекты метаданных подсистем **Обмен данными** и **Обмен данными в модели сервиса**.

Если для объекта метаданных требуется ограничить миграцию элементов при обмене, то для этого объекта необходимо создать правила регистрации при помощи конфигурации **Конвертация данных**. Если правило регистрации для объекта метаданных не создано, то считается, что миграция объектов такого типа не ограничена, даже если признак авторегистрации установлен в значение `Запретить`.

Рекомендации по включению объектов других подсистем библиотеки в состав планов обмена приведены в разделах **Использование при разработке конфигурации**, **Настройка обмена данными** соответствующих подсистем.

Модуль менеджера плана обмена

Для реализации прикладных задач зачастую недостаточно платформенных свойств и методов планов обмена, поэтому подсистема **Обмен данными** предоставляет дополнительные возможности, позволяющие использовать дополнительные свойства и методы планов обмена. Использование данных возможностей предполагает определение в модуле менеджера плана обмена специальных процедур и функций, одна из которых является обязательной (см. далее).

Обязательная процедура «ПриПолученииНастроек»

Для всех планов обмена, объявленных в процедуре `ПолучитьПланыОбмена` общего модуля `ОбменДаннымиПереопределяемый` (см. раздел Разработка планов обмена), в модуле менеджера должна быть обязательно объявлена экспортная процедура `ПриПолученииНастроек`. Данная процедура представляет собой «точку входа», позволяющую механизмам подсистемы **Обмен данными** получить доступ ко всем свойствам и методам плана обмена, отвечающим за технологическую и прикладную специфику конкретного обмена.

Объявление:

```
Процедура ПриПолученииНастроек(Настройки) Экспорт
```

Параметры:

- `Настройки`, тип - `Структура`, `Ключ` – имя настройки, `Значение` – значение настройки:

Имя настройки (тип значения)	Описание
`Алгоритмы` (`Структура`)	Управление доступностью методов плана обмена (экспортных процедур и функций), поддерживаемых подсистемой `Обмен данными` для планов обмена. `Ключ` – имя метода (процедуры/функции), используется только для чтения. `Значение` – `Булево` – признак использования метода. `Истина` – метод используется в данном плане обмена, `Ложь` – метод не используется. Значение по умолчанию для всех методов – `Ложь`. Список методов: ПриПолученииВариантовНастроекОбмена ПриПолученииОписанияВариантаНастройки ОбработчикПроверкиОграниченийПередачиДанных ОбработчикПроверкиЗначенийПоУмолчанию ОбработчикПроверкиПараметровУчета ПриПодключенииККорреспонденту ПриОтправкеДанныхОтправителя ПриПолученииДанныхОтправителя НастроитьИнтерактивнуюВыгрузку НастроитьИнтерактивнуюВыгрузкуВМоделиСервиса ОписаниеОграниченийПередачиДанных ОписаниеЗначенийПоУмолчанию ПредставлениеОтбораИнтерактивнойВыгрузки ПриСохраненииНастроекСинхронизацииДанных ПриОпределенииПоддерживаемыхОбъектовФормата ПриОпределенииПоддерживаемыхКорреспондентомОбъектовФормата Для того чтобы использовать процедуру, необходимо установить значение `Истина` для значения соответствующего ключа структуры, а также объявить соответствующую экспортную процедуру в модуле менеджера плана обмена. Примеры реализации см. в демонстрационной конфигурации в планах обмена `_ДемоОбменВРаспределеннойИнформационнойБазе` и `_ДемоАвтономнаяРабота`.
`НазначениеПланаОбмена` (`Строка`)	Свойство, определяющее принадлежность плана обмена к одной из нескольких категорий, влияет на его использование в различных механизмах подсистемы **Обмен данными**: • `РИБ` – обмен РИБ без использования ограничений миграции данных, • `РИБСФильтром` – обмен РИБ с использованием ограничений миграции данных, • `СинхронизацияСДругойПрограммой` – обмен с другой программой. В большинстве случаев достаточно использовать значения по умолчанию: для планов обмена РИБ определяется как `РИБ`, для остальных – `СинхронизацияСДругойПрограммой`.
`ЭтоПланОбменаXDTO` (`Булево`)	Признак того, что план обмена используется для `ОУФ`. Значение по умолчанию – `Ложь`.
`ФорматОбмена` (`Строка`)	Имя формата обмена, используемого для данного плана обмена. Используется только для `ОУФ` (см. свойство `ЭтоПланОбменаXDTO`).
`РасширенияФорматаОбмена` (`Соответствие`)	Соответствие URI пространства имен схемы расширения формата номеру расширяемой версии формата. Используется только для `ОУФ` (см. свойство `ЭтоПланОбменаXDTO`).
`ВерсииФорматаОбмена` (`Соответствие`)	Соответствие номеров поддерживаемых версий формата данных и ссылок на общие модули, реализующих логику обмена через конкретную версию формата. Используется только для `ОУФ` (см. свойство `ЭтоПланОбменаXDTO`).
`ПланОбменаИспользуетсяВМоделиСервиса` (`Булево`)	Признак использования плана обмена для организации обмена в модели сервиса. Если признак установлен, то в сервисе можно включить обмен данными с использованием этого плана обмена, а также можно использовать этот план обмена для организации автономной работы в модели сервиса. Если признак не установлен, то план обмена будет использоваться только для обмена в локальном режиме работы конфигурации. Значение по умолчанию – `Ложь`.
`ИмяКонфигурацииИсточника` (`Строка`)	Строковый идентификатор, отличающий эту конфигурацию от других при работе в модели сервиса. Заполняется только для плана обмена, используемого в режиме сервиса (см. свойство `ПланОбменаИспользуетсяВМоделиСервиса`). Например, для конфигурации **Бухгалтерия предприятия** значение свойства будет `БухгалтерияПредприятия`.
`ИмяКонфигурацииПриемника` (`Структура`)	Элементы структуры определяют перечень конфигураций, обмен с которыми возможен через данный план обмена. Ключ элемента структуры – строковый идентификатор конфигурации. Должен совпадать со значением свойства `ИмяКонфигурацииИсточника` соответствующего плана обмена

Имя настройки (тип значения)	Описание
	конфигурации-корреспондента. Значение элемента структуры = `Неопределено`. Зарезервировано для дальнейшего использования. Например, если необходимо указать, что текущий план обмена используется для обмена с конфигурацией **Бухгалтерия предприятия**, в данную структуру необходимо вставить элемент с ключом `БухгалтерияПредприятия`. Свойство обязательно для заполнения для планов обмена, удовлетворяющих всем перечисленным условиям: • план обмена используется в режиме сервиса (см. свойство `ПланОбменаИспользуетсяВМоделиСервиса`); • план обмена не используется в `ОУФ` (см. свойство `ЭтоПланОбменаXDTO`); • план обмена не используется для обмена в РИБ.
`ПредупреждатьОНесоответствииВерсийПланаОбмена` (`Булево`)	Признак необходимости проверки на расхождение версий в правилах конвертации. Проверка выполняется при загрузке комплекта правил, при отправке и получении данных. Используется только для `УОП`.
`ИмяПланаОбменаДляПереходаНаНовыйОбмен` (`Строка`)	При выполнении перехода от использования одного плана обмена к использованию другого плана обмена (например, при переходе с `УОП` на аналогичный по функциональности `ОУФ`) определяет имя плана обмена – приемника. Если свойство установлено, в рабочих местах управления настройками не будет предлагаться настроить этот вид обмена. Существующие обмены этого вида будут продолжать отображаться в списке настроенных обменов. По умолчанию значение не заполнено.
`ВариантыНастроекОбмена` (`ТаблицаЗначений`)	Определяет набор предопределенных вариантов настроек на плане обмена. Подробнее см. Варианты настроек.
`ПравилаРегистрацииВМенеджере` (`Булево`)	Признак использования правил регистрации объектов, расположенных в общем модуле. Используется только для ОУФ (см. свойство `ЭтоПланОбменаXDTO`).
`МенеджерРегистрации` (`Строка`)	Имя общего модуля, в котором располагаются правила регистрации объектов. Используется совместно с `ПравилаРегистрацииВМенеджере`.

Варианты настроек

Подсистема **Обмен данными** поддерживает механизм предопределенных вариантов настроек. Суть механизма заключается в том, что на одном плане обмена могут быть реализованы несколько видов обмена (далее – варианты настроек), отличающихся друг от друга определенным набором свойств (далее – свойства вариантов настроек), определяющих функциональность обмена данными на следующих участках:

- интерфейс основных операций обмена (настройка, выполнение и т.д.);
- логика обмена данными (параметризация логики конвертации, регистрации данных);
- прочая функциональность, реализованная в подсистеме **Обмен данными**.

Необходимость использования нескольких вариантов настроек возникает в следующих случаях:

- один план обмена используется для организации обмена с несколькими конфигурациями корреспондентов или несколькими разными приложениями (пример: один план обмена ОУФ может быть использован для обмена с конфигурациями **Бухгалтерия предприятия** и **Управление торговлей**);
- один план обмена используется для организации обмена в нескольких режимах, каждый из которых представляется пользователям как самостоятельный вид синхронизации данных (пример: **Отправка данных**, **Получение данных**, **Двусторонний**, см. плана обмена `_ДемоОбменСБиблиотекойСтандартныхПодсистем` в демонстрационной конфигурации).

Для плана обмена рекомендуется определить хотя бы один вариант настройки (вариант по умолчанию), параметры которого будут использоваться для построения дерева доступных для настройки видов синхронизации.

Определение перечня всех доступных вариантов настроек выполняется в процедуре `ПриПолученииВариантовНастроекОбмена` модуля менеджера плана обмена (см. ниже).

Определение свойств, характерных для варианта настройки, выполняется в процедуре `ПриПолученииОписанияВариантаНастройки` (см. ниже).

Процедура «ПриПолученииВариантовНастроекОбмена»

Применяется для определения перечня доступных вариантов настроек плана обмена, а также позволяет задать признаки их использования для обмена с корреспондентами, работающими в локальном режиме и в режиме сервиса.

Объявление:

```
Процедура ПриПолученииВариантовНастроекОбмена(ВариантыНастроекОбмена, ПараметрыКонтекста) Экспорт
```

Параметры:

- `ВариантыНастроекОбмена`, тип - `ТаблицаЗначений`. Добавление варианта происходит путем добавления новой строки и заполнения следующих колонок:

Колонка (тип значения)	Описание
`ИдентификаторНастройки` (`Строка`)	Строковый идентификатор варианта настройки, отличающий его от других вариантов настроек данного плана обмена.
`КорреспондентВМоделиСервиса` (`Булево`)	Признак того, что план обмена поддерживает обмен данными с корреспондентом, работающим в модели сервиса.
`КорреспондентВЛокальномРежиме` (`Булево`)	Признак того, что план обмена поддерживает обмен данными с корреспондентом, работающим в локальном режиме.

- `ПараметрыКонтекста`, тип - `Структура`. Определяет режим работы метода через контекст его выполнения. `Ключ` – параметр контекста, `Значение` – его значение:

Параметр контекста (тип значения)	Описание
`ИмяКорреспондента` (`Строка`, `Неопределено`)	Строковый идентификатор конфигурации или приложения, с которым настраивается обмен (см. свойство `ИмяКонфигурацииИсточника` метода `ПриПолученииНастроек`). Незаполненное значение параметра означает отсутствие информации о корреспонденте.
`ВерсияКорреспондента` (`Строка`)	Номер версии корреспондента. Незаполненное значение параметра означает отсутствие информации о корреспонденте.
`КорреспондентВМоделиСервиса` (`Булево`)	Признак того, что план корреспондент работает в модели сервиса.

Если процедура `ПриПолученииВариантовНастроекОбмена` не определена, подсистема **Обмен данными** автоматически создает служебный вариант настройки со следующими значениями свойств:

- `ИдентификаторНастройки` – пустая строка;
- `КорреспондентВМоделиСервиса` – если план обмена поддерживает работу в модели сервиса и это модель сервиса – `Истина`, иначе – `Ложь`;
- `КорреспондентВЛокальномРежиме` – `Истина`.

Процедура «ПриПолученииОписанияВариантаНастройки»

Применяется, если необходимо переопределить свойства варианта настройки.

Объявление:

```
Процедура ПриПолученииОписанияВариантаНастройки(ОписаниеВарианта, ИдентификаторНастройки, ПараметрыКонтекста) Экспорт
```

Параметры:

Колонка (тип значения)	Описание
`ОписаниеВарианта` (`Структура`)	Содержит перечень доступных свойств варианта настройки. Подробнее см. ниже.
`ИдентификаторНастройки` (`Строка`)	Строковый идентификатор варианта настройки, отличающий его от других вариантов настроек данного плана обмена.
`ПараметрыКонтекста` (`Структура`)	Определяет режим работы метода через контекст его выполнения. Ключ – параметр контекста, значение – его значение. Подробнее см. ниже.

Параметр `ОписаниеВарианта`:

Имя свойства (тип значения)	Описание
`ИспользоватьПомощникСозданияОбменаДанными` (`Булево`)	Определяет, будет ли использоваться помощник для создания новых узлов плана обмена. Значение по умолчанию – `Истина`.
`ИмяФормыПомощникаНастройкиСинхронизацииДанных` (`Строка`)	Полное имя формы, которая будет открыта для настройки правил отправки и получения данных. По умолчанию свойство не заполнено (используется форма узла плана обмена).
`ЗаголовокКомандыДляСозданияНовогоОбменаДанными` (`Строка`)	Представление команды, выводимое в пользовательском интерфейсе при создании новой настройки обмена данными. Значение по умолчанию – синоним плана обмена.
`ЗаголовокПомощникаСозданияОбмена` (`Строка`)	Представление заголовка формы помощника создания обмена данными, выводимое в пользовательском интерфейсе. Значение по умолчанию – строка, образованная по шаблону **Синхронизация данных с [Название программы] (настройка)**, где **[Название программы]** – синоним плана обмена.
`ЗаголовокУзлаПланаОбмена` (`Строка`)	Представление узла плана обмена, выводимое в пользовательском интерфейсе. Значение по умолчанию – синоним плана обмена.
`ИмяКонфигурацииКорреспондента` (`Строка`)	Идентификатор конфигурации-корреспондента, с которой настраивается обмен по текущему варианту. Используется для группировки в дереве доступных синхронизаций данных. Рекомендуется заполнять в том случае, когда возможна настройка обмена с одной и той же конфигурацией через разные планы обмена – например, настройка обмена с конфигураций **Управление нашей фирмой** по планам обмена УОП и ОУФ. По умолчанию свойство не заполнено.
`НаименованиеКонфигурацииКорреспондента` (`Строка`)	Представление конфигурации корреспондента, выводимое в пользовательском интерфейсе. Значение по умолчанию – пустая строка.
`КраткаяИнформацияПоОбмену` (`Строка`)	Краткое описание обмена данными, которое выводится на странице помощника настройки новой синхронизации. По умолчанию свойство не заполнено.
`ПодробнаяИнформацияПоОбмену` (`Строка`)	Ссылка на веб-страницу или полный путь к форме внутри конфигурации строкой. По умолчанию свойство не заполнено.
`ИмяФайлаНастроекДляПриемника` (`Строка`)	Имя файла настроек по умолчанию. В этот файл будут выгружены настройки обмена для приемника. Свойство используется только для планов обмена `ОУФ` и `УОП`. По умолчанию свойство не заполнено (имя файла необходимо будет указать вручную).
`ПутьКФайлуКомплектаПравилНаПользовательскомСайте` (`Строка`)	Путь к файлу комплекта правил в виде архива на пользовательском сайте в разделе конфигурации. По умолчанию свойство не заполнено (путь необходимо будет указать вручную).
`ПутьКФайлуКомплектаПравилВКаталогеШаблонов` (`Строка`)	Относительный путь к файлу комплекта правил в каталоге шаблонов **1С:Предприятия**. По умолчанию свойство не заполнено (путь необходимо будет указать вручную).
`ИспользуемыеТранспортыСообщенийОбмена` (`Массив`)	Перечень используемых транспортов сообщений для текущего плана обмена. Если не заполнен, будут доступны все допустимые настройки транспорта. Значение по умолчанию – пустой `Массив`.
`ИмяФормыСозданияНачальногоОбраза` (`Строка`)	Имя формы плана обмена, используемой для создания начального образа. Используется только для планов обмена `РИБ`. По умолчанию свойство не заполнено (используется стандартная форма).
`ОбщиеДанныеУзлов` (`Строка`)	Имена реквизитов и табличных частей плана обмена, которые являются общими для пары обменивающихся конфигураций (см. раздел Общие данные узлов). Имена перечисляются через запятую. По умолчанию свойство не заполнено (нет общих данных).
`Отборы` (`Структура`)	Отборы для этого приложения, заполненные начальными данными. Имеет смысл только для планов обмена, используемых для организации автономной работы в модели сервиса. Если отборы не предусмотрены, не заполняется. Структура настроек должна повторять состав реквизитов шапки и табличных частей плана обмена, предназначенных для хранения отборов. Для реквизитов шапки используются аналогичные по ключу и значению элементы структуры, а для

Имя свойства (тип значения)	Описание
	табличных частей используются структуры, содержащие массивы значений полей табличных частей плана обмена. Значение по умолчанию – пустая `Структура`. Пример использования см. в демонстрационной конфигурации, модуль менеджера плана обмена `_ДемоАвтономнаяРабота`.
`ЗначенияПоУмолчанию` (`Структура`)	Значения по умолчанию для этого приложения, заполненные начальными данными. Имеет смысл только для планов обмена, используемых для организации автономной работы в модели сервиса. Структура настроек должна повторять состав реквизитов шапки плана обмена, предназначенных для хранения значений по умолчанию. Для реквизитов шапки используются аналогичные по ключу и значению элементы структуры. Значение по умолчанию – пустая `Структура`.
`ПояснениеДляНастройкиПараметровУчета` (`Строка`)	Пояснение о последовательности действий пользователя для настройки параметров учета в текущей информационной базе. По умолчанию свойство не заполнено.
`РасширениеФормата` (`Строка`)	Имя расширения формата, применимое для данного варианта настройки. Используется только для `ОУФ` (см. свойство `ЭтоПланОбменаXDTO`).

- `ПараметрыКонтекста`, тип - `Структура`. Определяет режим работы метода через контекст его выполнения. Ключ – параметр контекста, значение – его значение:

Параметр контекста (тип значения)	Описание
`ИмяКорреспондента` (`Строка`, `Неопределено`)	Строковый идентификатор конфигурации или приложения, с которым настраивается обмен (см. свойство `ИмяКонфигурацииИсточника` метода `ПриПолученииНастроек`).
`ВерсияКорреспондента` (`Строка`)	Номер версии корреспондента.

Процедура «ОбработчикПроверкиОграниченийПередачиДанных»

Обработчик вызывается перед началом формирования сообщения обмена для отправки данных корреспонденту. Используется только для планов обмена ОУФ. Позволяет проверить заполненность всех необходимых параметров отправки данных, без которых невозможно сформировать корректное сообщение обмена. Например, для корреспондента необходимо отправить `Договоры`, но на узле обмена не заполнено правило формирования договоров из **Заказов покупателей**. Если параметры отправки данных не настроены или настроены не полностью, то в обработчике необходимо переменной `Отказ` установить значение `Истина`. Дополнительно можно назначить переменной `СообщениеОбОшибке` описание, которое будет выведено пользователю.

Объявление:

```
Процедура ОбработчикПроверкиОграниченийПередачиДанных(Отказ, ПараметрыОбработчика, СообщениеОбОшибке = "") Экспорт
```

Параметры:

- `Отказ`. Тип – `Булево`. Если в информационной базе заданы не все настройки параметров, необходимых для корректной отправки данных, то нужно установить данному флагу значение `Истина`.
- `ПараметрыОбработчика`. Тип – `Структура`. Состав:
  - `Корреспондент`. Тип – `ПланОбменаОбъект`. Узел плана обмена, соответствующий корреспонденту;
  - `ПоддерживаемыеОбъектыXDTO`. Тип – `Массив`. Содержит перечень строковых идентификаторов объектов формата (например, `Справочник.Контрагенты`), которые могут быть получены корреспондентом и отправка которых поддерживается в этой информационной базе.
- `СообщениеОбОшибке`. Тип – `Строка`. Сообщение пользователю о действиях, которые необходимо выполнить для правильной настройки параметров отправки данных.

Процедура «ОбработчикПроверкиЗначенийПоУмолчанию»

Обработчик вызывается перед началом загрузки сообщения обмена, полученного от корреспондента. Используется только для планов обмена ОУФ. Позволяет проверить заполненность значений по умолчанию, используемых для подстановки в объекты информационной базы, создаваемые при загрузке сообщения обмена. Подробнее см. Использование значений по умолчанию в обмене данными.

Объявление:

```
Процедура ОбработчикПроверкиЗначенийПоУмолчанию(Отказ, ПараметрыОбработчика, СообщениеОбОшибке = "") Экспорт
```

Параметры:

- `Отказ`. Тип – `Булево`. Если в информационной базе заданы не все значения по умолчанию, необходимые для корректного заполнения данных при загрузке, то нужно установить данному флагу значение `Истина`.
- `ПараметрыОбработчика`. Тип – `Структура`. Состав:

  - `Корреспондент`. Тип – `ПланОбменаОбъект`. Узел плана обмена, соответствующий корреспонденту;
  - `ПоддерживаемыеОбъектыXDTO`. Тип – `Массив`. Содержит перечень объектов формата, которые могут быть отправлены корреспондентом и получение которых поддерживается в этой информационной базе.
- `СообщениеОбОшибке`. Тип – `Строка`. Сообщение пользователю о действиях, которые необходимо выполнить для правильной настройки значений по умолчанию.

Процедура «ОбработчикПроверкиПараметровУчета»

Обработчик вызывается помощником настройки обмена при настройке обмена данными через Интернет. Позволяет определить алгоритм проверки параметров учета в конфигурации. Если в результате проверки выяснится, что параметры учета не настроены или настроены не полностью, то в обработчике необходимо переменной `Отказ` установить значение `Истина`. Дополнительно можно назначить переменной `Сообщение` описание для пользователя, какие именно настройки необходимо заполнить.

Объявление:

```
Процедура ОбработчикПроверкиПараметровУчета(Отказ, Получатель, Сообщение) Экспорт
```

Параметры:

- `Отказ`. Тип – `Булево`. Если в информационной базе заданы не все настройки параметров учета, необходимые для правильной работы обмена данными, то нужно установить этот флажок;
- `Получатель`. Тип – `ПланОбменаСсылка`. Ссылка на узел плана обмена, для которого выполняется настройка обмена данными;
- `Сообщение`. Тип – `Строка`. Сообщение пользователю о действиях, которые необходимо выполнить для правильной настройки параметров учета в конфигурации.

Процедура «ПриПодключенииККорреспонденту»

Обработчик события при подключении к корреспонденту. Событие возникает при успешном подключении к корреспонденту и получении версии конфигурации корреспондента при настройке обмена с использованием помощника через прямое подключение или при подключении к корреспонденту через Интернет. В обработчике можно проанализировать версию корреспондента и, если настройка обмена не поддерживается с корреспондентом указанной версии, вызвать исключение.

Объявление:

```
Процедура ПриПодключенииККорреспонденту(ВерсияКорреспондента) Экспорт
```

Параметры:

- `ВерсияКорреспондента` (только чтение) – `Строка`. Версия конфигурации корреспондента, например «2.1.5.1».

Процедура «ПриОтправкеДанныхОтправителя»

Обработчик события при отправке данных узла-отправителя. Событие возникает при отправке данных узла-отправителя из текущей базы в базу-корреспондент, до помещения данных узла в сообщение обмена. В обработчике можно изменить отправляемые данные или вовсе отказаться от отправки данных узла.

Объявление:

```
Процедура ПриОтправкеДанныхОтправителя(Отправитель, Игнорировать) Экспорт
```

Параметры:

- `Отправитель`. Тип значения зависит от текущего состояния обмена данными с другой программой:
  - при настройке нового обмена данными – `Структура`. Сдержит настройки отборов на узле;
  - при выполнении обмена данными – `ПланОбменаОбъект`. Узел плана обмена, от имени которого выполняется отправка данных.
- `Игнорировать`. Тип – `Булево`. Признак отказа от выгрузки данных узла. Если в обработчике установить значение этого параметра в значение `Истина`, то отправка данных узла выполнена не будет. Значение по умолчанию – `Ложь`. При настройке нового обмена параметр не используется.

Процедура «ПриПолученииДанныхОтправителя»

Обработчик события при получении данных узла-отправителя. Событие возникает при получении данных узла-отправителя, когда данные узла прочитаны из сообщения обмена, но не записаны в информационную базу. В обработчике можно изменить полученные данные или вовсе отказаться от получения данных узла.

Объявление:

```
Процедура ПриПолученииДанныхОтправителя(Отправитель, Игнорировать) Экспорт
```

Параметры:

- `Отправитель`. Тип значения зависит от текущего состояния обмена данными с другой программой:
  - при настройке нового обмена данными – `Структура`. Содержит настройки отборов на узле;
  - при выполнении обмена данными – `ПланОбменаОбъект`. Узел плана обмена, от имени которого выполняется отправка данных.

- `Игнорировать`. Тип – `Булево`. Признак отказа от выгрузки данных узла. Если в обработчике установить значение этого параметра в значение `Истина`, то отправка данных узла выполнена не будет. Значение по умолчанию – `Ложь`. При настройке нового обмена параметр не используется.

Процедура «НастроитьИнтерактивнуюВыгрузку»

Применяется для всех видов обмена, кроме `РИБ`.

Предназначена для настройки параметров работы помощника обмена данными на этапе интерактивного дополнения выборки выгружаемых данных. Для настройки доступны три типовых варианта добавления данных к выгрузке (**Не добавлять**, **Добавить все документы за период**, **Добавить данные с произвольным отбором**) и один дополнительный вариант по переопределяемому сценарию узла. Каждый вариант можно отключить, изменить название, подсказку и т. д. При отключении всех вариантов этап интерактивного дополнения выборки выгружаемых данных будет целиком пропущен.

Подробное описание и пример использования см. в демонстрационной конфигурации, модуль менеджера плана обмена `_ДемоОбменСБиблиотекойСтандартныхПодсистем`.

Объявление:

```
Процедура НастроитьИнтерактивнуюВыгрузку(Получатель, Параметры) Экспорт
```

Параметры:

- `Получатель`. Тип – `ПланОбменаСсылка`. Узел, для которого производится настройка;
- `Параметры`. Тип – `Структура`. Параметры для изменения. Для настройки необходимо установить свойствам необходимые значения.

Процедура «НастроитьИнтерактивнуюВыгрузкуВМоделиСервиса»

Применяется для обменов, работающих в модели сервиса.

Аналог процедуры НастроитьИнтерактивнуюВыгрузку для работы в модели сервиса.

Подробное описание и пример использования см. в демонстрационной конфигурации, модуль менеджера плана обмена `_ДемоОбменСБиблиотекойСтандартныхПодсистем`.

Объявление:

```
Процедура НастроитьИнтерактивнуюВыгрузкуВМоделиСервиса(Получатель, Параметры) Экспорт
```

Параметры: см. описание для функции `НастроитьИнтерактивнуюВыгрузку`.

Функция «ОписаниеОграниченийПередачиДанных»

Применяется, если предусмотрены отборы. Имеет смысл только для планов обмена, используемых для организации автономной работы в модели сервиса.

Возвращает строку описания ограничений миграции данных для пользователя. Прикладной разработчик на основе установленных отборов на узле должен сформировать строку описания ограничений, удобную для восприятия пользователем.

Объявление:

```
Функция ОписаниеОграниченийПередачиДанных(НастройкаОтборовНаУзле, ВерсияКорреспондента, ИдентификаторНастройки) Экспорт
```

Параметры:

- `НастройкаОтборовНаУзле`. Тип – `Структура`. См. описание свойства варианта настроек `Отборы` в разделе Процедура «ПриПолученииОписанияВариантаНастройки»;
- `ВерсияКорреспондента`. Тип – `Строка`. Номер версии корреспондента.
- `ИдентификаторНастройки`. Тип – `Строка`. Идентификатор варианта настройки, для которого формируется описание.

Функция «ОписаниеЗначенийПоУмолчанию»

Применяется, если в обмене предусмотрены значения по умолчанию. Имеет смысл только для планов обмена, используемых для организации работы в модели сервиса.

Возвращает пользовательское представление значений по умолчанию в виде строки.

Объявление:

```
Функция ОписаниеЗначенийПоУмолчанию(ЗначенияПоУмолчаниюНаУзле, ВерсияКорреспондента, ИдентификаторНастройки) Экспорт
```

Параметры:

- `ЗначенияПоУмолчаниюНаУзле`. Тип – `Структура`. См. описание свойства варианта настроек `ЗначенияПоУмолчанию` в разделе Процедура «ПриПолученииОписанияВариантаНастройки»;
- `ВерсияКорреспондента`. Тип – `Строка`. Номер версии корреспондента;

- `ИдентификаторНастройки`. Тип – `Строка`. Идентификатор варианта настройки, для которого формируется описание.

Функция «ПредставлениеОтбораИнтерактивнойВыгрузки»

Применяется для всех видов обмена, кроме `РИБ`.

Возвращает представление отбора для варианта дополнения выгрузки по сценарию узла.

Объявление:

```
Функция ПредставлениеОтбораИнтерактивнойВыгрузки(Получатель, Параметры) Экспорт
```

Параметры:

- `Получатель`. Тип – `ПланОбменаСсылка`. Узел, для которого запрашивается представление.
- `Параметры`. Тип – `Структура`. Описывает текущие параметры отбора, для которых запрашивается представление, поля структуры соответствуют аналогичным полям, заполняемым в процедуре `НастроитьИнтерактивнуюВыгрузку`. Подробное описание и пример использования см. в демонстрационной конфигурации, модуль менеджера плана обмена `_ДемоОбменСБиблиотекойСтандартныхПодсистем`.

Процедура «ПриСохраненииНастроекСинхронизацииДанных»

Обработчик заполнения настроек отправки и получения данных на узле обмена. Вызывается при условии использования программного интерфейса `ОбменДаннымиСервер.ПриНачалеСохраненияНастроекСинхронизации()` из переопределяемого помощника настройки синхронизации данных.

Пример использования см. в демонстрационной конфигурации, модуль менеджера плана обмена `_ДемоОбменВРаспределеннойИнформационнойБазе` и форма `ПомощникНастройкиСинхронизацииДанных`.

Объявление:

```
Процедура ПриСохраненииНастроекСинхронизацииДанных(Корреспондент, ДанныеЗаполнения) Экспорт
```

Параметры:

- `Корреспондент`. Тип – `ПланОбменаОбъект`. Узел плана обмена, соответствующий корреспонденту.
- `ДанныеЗаполнения`. Тип – `Структура`. Произвольная структура с данными для заполнения параметров отправки и получения данных, переданная из помощника настройки синхронизации данных.

Процедура «ПриОпределенииПоддерживаемыхОбъектовФормата»

Обработчик предоставляет возможность переопределения состава поддерживаемых объектов формата. Используется только для планов обмена ОУФ. Может использоваться в случаях, когда необходимо ограничить список поддерживаемых объектов в зависимости от настроек информационной базы либо от контекста (от конкретного узла плана обмена). Вызывается механизмами обмена данными XDTO перед созданием и получением сообщения обмена и используется для последующей фильтрации выгружаемых и загружаемых данных. В параметре `ПоддерживаемыеОбъекты` на момент вызова обработчика содержится перечень поддерживаемых объектов, сформированный на основании информации о типах источников и приемников в правилах обработки данных (`ПОД`) и правилах конвертации объектов (`ПКО`).

Объявление:

```
Процедура ПриОпределенииПоддерживаемыхОбъектовФормата(ПоддерживаемыеОбъекты, Режим, УзелОбмена = Неопределено) Экспорт
```

Параметры:

- `ПоддерживаемыеОбъекты`. Тип – `ТаблицаЗначений`. Содержит перечень объектов формата, поддерживаемых данной информационной базой, в разрезе версий формата. Колонки:
  - `Версия`. Тип – `Строка`. Версия формата, например **1.6**;
  - `Объект`. Тип – `Строка`. Объект формата, например `Справочник.Номенклатура`;
  - `Отправка`. Тип – `Булево`. `Истина`, если в информационной базе поддерживается отправка данного объекта формата. Колонка доступна, если `Режим` = `Отправка` или `ОтправкаПолучение`;
  - `Получение`. Тип – `Булево`. `Истина`, если в информационной базе поддерживается получение данного объекта формата. Колонка доступна, если Режим = `Получение`, `ОтправкаПолучение`;
  - `Режим`. Тип – `Строка`. Вид запрашиваемой информации. Возможные значения: `Отправка` – запрос объектов формата, для которых поддерживается отправка; `Получение` – запрос объектов, для которых поддерживается получение; `ОтправкаПолучение` – запрос всех поддерживаемых объектов формата.
- `УзелОбмена`. Тип – `ПланОбменаСсылка`, `Неопределено`. Ссылка на узел плана обмена, соответствующий корреспонденту, в контексте которого запрашивается информация о поддерживаемых объектах формата. Может использоваться при необходимости ограничить список поддерживаемых этой информационной базой объектов в зависимости от варианта настройки обмена, заданного на узле.

Процедура «ПриОпределенииПоддерживаемыхКорреспондентомОбъектовФормата»

Обработчик предоставляет возможность переопределения состава поддерживаемых корреспондентом объектов формата. Используется только для планов обмена ОУФ. Может использоваться в случаях, когда необходимо дополнить или ограничить список поддерживаемых объектов, которые могут участвовать в обмене. Вызывается механизмами обмена данными XDTO перед созданием и получением сообщения обмена и используется для последующей фильтрации выгружаемых и загружаемых данных. В параметре `ПоддерживаемыеОбъекты` на момент вызова обработчика содержится перечень поддерживаемых объектов, полученный от корреспондента.

Объявление:

```
Процедура ПриОпределенииПоддерживаемыхКорреспондентомОбъектовФормата(УзелОбмена, ПоддерживаемыеОбъекты, Режим) Экспорт
```

Параметры:

- `УзелОбмена`. Тип – `ПланОбменаСсылка`. Узел плана обмена, соответствующий корреспонденту.
- `ПоддерживаемыеОбъекты`. Тип – `ТаблицаЗначений`. Содержит перечень объектов формата, поддерживаемых корреспондентом, в разрезе версий формата. Колонки:
  - `Версия`. Тип – `Строка`. Версия формата, например **1.6**;
  - `Объект`. Тип – `Строка`. Объект формата, например `Справочник.Номенклатура`;
  - `Отправка`. Тип – `Булево`. `Истина`, если в информационной базе поддерживается отправка данного объекта формата. Колонка доступна, если `Режим` = `Отправка` или `ОтправкаПолучение`;
  - `Получение`. Тип – `Булево`. `Истина`, если в информационной базе поддерживается получение данного объекта формата. Колонка доступна, если `Режим` = `Получение`, `ОтправкаПолучение`.
- `Режим`. Тип – `Строка`. Вид запрашиваемой информации. Возможные значения: `Отправка` – запрос объектов формата, для которых поддерживается отправка; `Получение` – запрос объектов, для которых поддерживается получение; `ОтправкаПолучение` – запрос всех поддерживаемых объектов формата.

Процедура «ВыполнитьПереходНаНовыйОбмен»

Процедура выполняет попытку перехода на новый обмен с существующего обмена. Используется только для планов обмена ОУФ. Вызывается, если в ходе синхронизации данных получено сообщение не соответствующее формату "старого" обмена. В процедуре описывается алгоритм создания (обновления) настройки синхронизации, перенос зарегистрированных, но еще не отправленных данных, корректировка сценариев обмена, перенос публичных идентификаторов. В случае успешного выполнения указанных действий, позволяет описать очистку зарегистрированных данных на старом узле информационной базы.

Объявление:

```
Процедура ВыполнитьПереходНаНовыйОбмен(НастройкаСинхронизацииДанных, ОбменЧерезВнешнееСоединение = Ложь) Экспорт
```

Параметры:

- `НастройкаСинхронизацииДанных`. Тип – `ПланОбменаСсылка`. Узел плана обмена, с которого выполняется переход на новый обмен.
- `ОбменЧерезВнешнееСоединение`. Тип – `Булево`. признак того что текущий обмен выполняется через внешнее соединение.

Функция «ПереходНаСинхронизациюЧерезУниверсальныйФорматВнешнееСоединение»

Вызывается при первом обмене по настройке синхронизации данных через универсальный формат через COM-соединение. Используется только для планов обмена ОУФ. Вызывается, если в ходе синхронизации данных получено сообщение не соответствующее формату "старого" обмена. В методе описывается алгоритм создания (обновления) настройки синхронизации, перенос зарегистрированных, но еще не отправленных данных, корректировка сценариев обмена, перенос публичных идентификаторов. В случае успешного выполнения указанных действий, позволяет описать очистку зарегистрированных данных на старом узле информационной базы.

Возвращаемое значение:

- `ПланОбменаСсылка`, `Неопределено`. При успешном выполнении перехода `ПланОбменаСсылка` на настройку синхронизации данных, а в случае непредвиденного завершения `Неопределено`.

Объявление:

```
Функция ПереходНаСинхронизациюЧерезУниверсальныйФорматВнешнееСоединение(ПараметрыНастройкиСинхронизацииДанных) Экспорт
```

Параметры:

- `ПараметрыНастройкиСинхронизацииДанных`. Тип – `Структура`. сведения о настройке синхронизации, с которой происходит переход:
  - `Код`. Тип – `Строка`. Код настройки;
  - `ВариантНастройки`. Тип – `Строка`. вариант настройки синхронизации данных через универсальный формат базы-корреспондента;
  - `Ошибка`. Тип – `Булево`. Признак наличия ошибки при выполнении функции;
  - `СообщениеОбОшибке`. Тип – `Строка`. Текст сообщения об ошибке.

Функция «ПереходНаСинхронизациюЧерезУниверсальныйФорматИнтернет»

Вызывается при первом обмене по настройке синхронизации данных в универсальном формате через WS-соединение. Используется только для планов обмена ОУФ и актуальна, когда корреспондент выполнил переход на универсальный формат, а текущая ИБ нет. В методе описывается алгоритм создания (обновления) настройки синхронизации, перенос зарегистрированных, но еще не отправленных данных, корректировка сценариев обмена, перенос публичных идентификаторов. В случае успешного выполнения указанных действий, позволяет описать очистку зарегистрированных данных на старом узле информационной базы.

Возвращаемое значение:

- `ПланОбменаСсылка`, `Неопределено`. При успешном выполнении перехода `ПланОбменаСсылка` на настройку синхронизации данных, а в случае непредвиденного завершения `Неопределено`.

Объявление:

```
Функция ПереходНаСинхронизациюЧерезУниверсальныйФорматИнтернет(КодУзла, Ошибка) Экспорт
```

Параметры:

- `КодУзла`. Тип – `Строка`. Код настройки синхронизации (код узла обмена).
- `Ошибка`. Тип – `Булево`. Признак наличия ошибки при выполнении функции.

Режим совместимости для планов обмена УОП

Этот режим предназначен для поддержания обратной совместимости обмена данными между конфигурациями на базе БСП версии 2.0 и ниже – с одной стороны и конфигурациями на базе БСП 2.1 и выше – с другой.

Для конфигураций, рассчитанных только на локальный режим работы, необходимо принять решение по поводу включения режима совместимости.

Для конфигураций, рассчитанных на работу в модели сервиса, режим совместимости отключается принудительно (его нельзя включить).

По умолчанию для новых правил конвертации режим совместимости с конфигурациями на базе БСП версии 2.0 и более ранних отключен. При необходимости поддержать обмен данными с такими конфигурациями следует установить режим совместимости в значение **Версия БСП 2.0**. Режим совместимости устанавливается в конфигурации **Конвертация данных** версии 2.1.6 и выше в свойствах конвертации на закладке `Дополнительно`. Значение сохраняется в правилах обмена.

Для существующих правил конвертации по умолчанию режим совместимости установлен в значение **Версия БСП 2.0**.

При включенном режиме совместимости становится недоступным режим отладки, а также режим безопасного выполнения кода обработчиков.

В таблице приведены варианты использования режимов обмена данными.

Режим совместимости	Режим отладки	Выполнение кода обработчиков
Отключен [1]	Отключен [1]	Код обработчиков выгрузки и загрузки выполняется из обработок в составе конфигурации
Отключен [1]	Включен	Код обработчиков выгрузки и загрузки выполняется из внешних обработок
Включен	Недоступен (Отключен)	Код обработчиков выполняется: • при загрузке – из файла сообщения обмена, • при выгрузке – из правил обмена

> [1] Режимы совместимости и отладки всегда отключены в модели сервиса.

Безопасное выполнение кода обработчиков для планов обмена УОП

При отключенных режимах совместимости и отладки код обработчиков выгрузки и загрузки выполняется из обработок в составе конфигурации. Это обеспечивает высокий уровень безопасности обмена данными, прежде всего в модели сервиса.

При включенном режиме совместимости код обработчиков загрузки зачитывается и выполняется из сообщения обмена данными, что является небезопасным.

При отключенном режиме совместимости необходимо:

- В инструменте **Конвертация данных** версии 2.1.6 и выше:
  - отключить режим совместимости в свойствах конвертации;
  - сохранить правила конвертации объектов, установить флажок сохранения кода обработчиков выгрузки в текстовый документ и указать путь к нему (при этом модуль загрузки необходимо сформировать из второго комплекта правил аналогичным образом).
- В конфигурации:
  - загрузить правила конвертации, сохраненные в **Конвертации данных**;
  - создать обработки в составе конфигурации. Рекомендуемые имена обработок: `ОбработчикиЗагрузкиИз<ИмяКонфигурации>` и `ОбработчикиВыгрузкиВ<ИмяКонфигурации>`;
  - добавить функции в модуль менеджера плана обмена (см. ниже);
  - скопировать текст модулей из выгруженных ранее файлов в модули созданных обработок.

Формы плана обмена

Формы узла и списка плана обмена создаются по усмотрению разработчика. Если для плана обмена предполагается использовать ограничение миграции данных по узлам, то для этого следует в форме узла предусмотреть редактирование настроек ограничения миграции данных.

В форме узла должны быть определены обработчики `ПриЗакрыти` и `ПриЗаписиНаСервере`:

```
&НаКлиенте
Процедура ПриЗакрытии()

    Оповестить("Запись_УзелПланаОбмена");

КонецПроцедуры
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

    ОбменДаннымиСервер.ФормаУзлаПриЗаписиНаСервере(ТекущийОбъект, Отказ);

КонецПроцедуры
```

Опционально в модуле формы можно определить обработчик `ПередЗаписью`. Это может понадобиться, если планируется частое изменение настроек узла плана обмена и зарегистрировано достаточно большое количество объектов. В этом случае запись будет выполнена в фоновом режиме.

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)

    ОбменДаннымиКлиент.ПередЗаписью(ЭтотОбъект, Отказ, ПараметрыЗаписи);

КонецПроцедуры
```

> **Важно:**
>
> Для использования `ОбменДаннымиКлиент.ПередЗаписью` необходимо, что бы реквизит формы **Объект** содержал все необходимые изменения и не изменялся в последующих обработчиках, таких как `ПередЗаписьюНаСервере`, `ПриЗаписи` и т.д.

Если предполагается использование формы узла для обмена данными в модели сервиса, то при работе в этом режиме рекомендуется скрывать от пользователя код и наименование узла.

Дополнительно необходимо разработать произвольные формы плана обмена. Эти формы используются на этапе настройки обмена данными помощником настройки.

Возможность использования форм для планов обмена в зависимости от вида обмена представлена в таблице ниже.

Имя формы узла плана обмена	УОП, УО, ОУФ	АРМ	РИБ
`ФормаНастройкиУзла`	–	+	–
`ФормаСозданияНачальногоОбраза`	–	–	+
`ФормаПомощникаНастройкиСинхронизацииДанных`	+	–	+

Разработка формы «ФормаНастройкиУзла»

Для планов обмена, используемых для организации автономной работы в модели сервиса, должна быть создана произвольная форма с предопределенным именем `ФормаНастройкиУзла`. Она вызывается помощником настройки обмена данными для задания ограничения миграции данных.

Состав реквизитов формы повторяет состав реквизитов шапки и табличных частей плана обмена. Для формы определены обязательные реквизиты формы: `НастройкаОтборовНаУзле` произвольного типа и `ВерсияКорреспондента` (`Строка`, неограниченной длины). Для реквизитов формы, которые размещены в форме и определяют признак модифицированности формы, следует установить свойство `Сохраняемые данные`.

В модуле формы обязательно наличие следующих обработчиков:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    ОбменДаннымиСервер.ФормаНастройкиУзлаПриСозданииНаСервере(ЭтотОбъект, "_ДемоОбменСБиблиотекойСтандартныхПодсистем");

КонецПроцедуры
&НаКлиенте
Процедура ПередЗакрытием(Отказ, СтандартнаяОбработка)

    ОбменДаннымиКлиент.ФормаНастройкиПередЗакрытием(Отказ, ЭтотОбъект);

КонецПроцедуры
&НаКлиенте
Процедура КомандаОК(Команда)

    ОбменДаннымиКлиент.ФормаНастройкиУзлаКомандаЗакрытьФорму(ЭтотОбъект);

КонецПроцедуры
```

В качестве второго параметра процедуры `ФормаНастройкиУзлаПриСозданииНаСервере` следует использовать имя плана обмена (`Строка`), как оно задано в конфигураторе. В свойствах формы для обработчиков событий `ПриСозданииНаСервере` и `ПередЗакрытием` необходимо задать

соответствующие процедуры модуля формы.

Если некоторые из реквизитов плана обмена являются обязательными для заполнения, то в форме настроек узла для этих реквизитов следует установить свойство **Проверка заполнения** в значение **Выдавать ошибку**. При создании обмена данными с использованием помощника система потребует заполнения этих реквизитов.

Пример реализации формы можно посмотреть в демонстрационной конфигурации в плане обмена `_ДемоАвтономнаяРабота`.

Разработка формы «ФормаСозданияНачальногоОбраза»

Дополнительно для плана обмена РИБ возможно определить форму создания первоначального образа подчиненного узла с произвольным именем. Эта форма не является обязательной. Она либо используется для вызова стандартного окна создания начального образа, либо может переопределять стандартный процесс создания первоначального образа. Имя формы должно быть указано в процедуре `ПриПолученииОписанияВариантовНастройки` модуля менеджера плана обмена для соответствующего варианта настройки.

Разработка формы «ФормаПомощникаНастройкиСинхронизацииДанных»

Для планов обмена УОП, УО, ОУФ и РИБ (кроме АРМ) можно создать произвольную дополнительную форму и определить ее в качестве помощника настройки правил отправки и получения данных вместо используемой по умолчанию формы узла. Эта форма не является обязательной. Имя формы не регламентируется. Для использования формы при настройке синхронизации ее имя следует указать в процедуре `ПриПолученииОписанияВариантовНастройки` модуля менеджера плана обмена для соответствующего варианта настройки.

В качестве обязательного параметра при открытии в форму передается `УзелОбмена`, тип `ПланОбменаСсылка` – содержит ссылку на настраиваемый узел, соответствующий корреспонденту. В форме необходимо реализовать заполнение и сохранение настроек синхронизации данных.

Пример реализации формы помощника с использованием программного интерфейса `ОбменДаннымиСервер.ПриНачалеСохраненияНастроекСинхронизации()` можно посмотреть в демонстрационной конфигурации в плане обмена `_ДемоОбменВРаспределеннойИнформационнойБазе`.

Настройка отправки данных в помощнике интерактивной выгрузки. Дополнительная форма настройки

При использовании помощника интерактивного обмена существует возможность программной настройки этапа отправки данных для включения в состав передаваемой информации объектов, определенных пользователем, например «все документы за прошлый месяц». При этом к данным пользователя автоматически применяются и общие ограничения узла, например отбор по организациям. По умолчанию предлагаются три варианта дополнения отправляемых данных: не добавлять, добавить документы с отбором за период, добавить произвольные данные (документы и НСИ).

Отключение, настройка предопределенных вариантов, изменение их порядка в форме, определение дополнительного сценария настройки производятся в процедуре `НастроитьИнтерактивнуюВыгрузку` менеджера плана обмена. Для настройки необходимо изменить параметры, предлагаемые по умолчанию. Также нужно создать форму для редактирования настроек отбора дополнительных данных по сценарию узла. Полное имя этой формы требуется указать в соответствующем поле параметров.

Эта форма будет открыта для выбора из помощника интерактивного обмена с установленными параметрами, описывающими текущий отбор. Форма должна вернуть результатом выбора структуру, описывающую отбор, измененный пользователем.

Состав параметров открытия, описание результата выбора и пример реализации настройки можно увидеть в демонстрационной конфигурации. В модуле менеджера плана обмена `_ДемоОбменСБиблиотекойСтандартныхПодсистем` демонстрируется настройка поведения помощника интерактивного обмена. В форме `НастройкаВыгрузки` этого плана обмена демонстрируются прием, обработка и обратная передача помощнику отредактированных параметров отбора.

Правила регистрации

Макет правил регистрации

План обмена может содержать только один макет правил регистрации. Наличие макета правил регистрации рекомендуется, но не является обязательным. Наличие макета свидетельствует о необходимости выполнять ограничение миграции данных при обмене.

Правила регистрации хранятся в текстовом макете, в который помещается информация из файла правил в формате XML. Исходный файл правил формируется при помощи конфигурации **Конвертация данных**, редакция 3.1. Кодировка файла: UTF-8.

Для размещения/обновления правил регистрации в макете необходимо:

- открыть файл правил XML в конфигурации как текстовый документ (с помощью команды меню **Файл** – **Открыть**);
- скопировать в буфер обмена содержимое файла;
- вставить в макет плана обмена данные из буфера обмена.

Имя макета правил регистрации для всех планов обмена строго регламентировано и должно принимать значение `ПравилаРегистрации`.

Менеджер регистрации

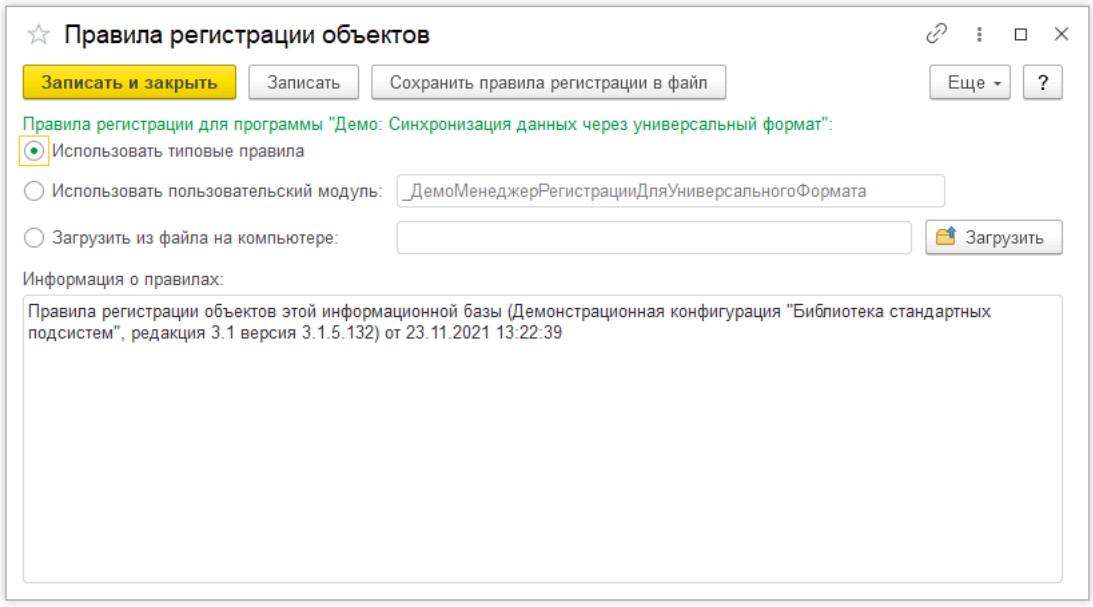
Для обменов данными через универсальный формат доступно использование менеджера регистрации. В этом случае код правил регистрации располагается в общем модуле, что позволяет отказаться от использования небезопасного метода `Выполнить`, позволяет производить отладку без дополнительных инструментов, использовать расширения конфигурации для удобной и быстрой кастомизации правил регистрации.

Для использования менеджера регистрации необходимо в модуле менеджера соответствующего плана обмена, в процедуре `ПриПолученииНастроек` заполнить настройки `ПравилаРегистрацииВМенеджере` и `МенеджерРегистрации`.

Пример заполнения настроек:

```
Процедура ПриПолученииНастроек(Настройки) Экспорт

    …

    Настройки.ПравилаРегистрацииВМенеджере = Истина;
    Настройки.МенеджерРегистрации = "_ДемоМенеджерРегистрацииДляУниверсальногоФормата";

КонецПроцедуры
```

Для создания общего модуля необходимо использовать конфигурацию **Конвертация данных** версии не ниже 3.1.1.6. У общего модуля должны быть установлены следующие контексты выполнения: **Сервер**, **Внешнее соединение**, **Клиент (обычное приложение)**.



Макеты правил обмена для планов обмена УОП

План обмена может содержать один или несколько макетов правил обмена. Наличие макетов правил обмена рекомендуется, но не является обязательным. Правила обмена хранятся в текстовом макете, в который помещается информация из файла правил обмена в формате XML. Исходный файл правил формируется при помощи конфигурации **Конвертация данных**, редакция 2.1. Кодировка файла: UTF-8.

Для размещения правил обмена в макете необходимо выполнить ту же последовательность действий, что и при добавлении правил регистрации (см. раздел Макет правил регистрации).

Имя макета правил обмена должно соответствовать шаблону: `ПравилаОбмена<Приемник><ВерсияПриемника>`, где `Приемник` – имя конфигурации-приемника, `ВерсияПриемника` – номер версии конфигурации-приемника.

Например, `ПравилаОбменаБухгалтерияПредприятия_1_6_18`.

Обязательным является наличие макета с именем `ПравилаОбмена`. Правила обмена из этого макета будут использоваться как правила обмена по умолчанию при настройке обмена данными с использованием помощника настройки.

Создание подписок на события

Для работы подсистемы обмена данными необходимо создать подписки на события. Подписки на события создаются для работы механизма регистрации изменения данных.

Механизм регистрации изменения данных используется не только для реализации прикладной логики ограничений миграции данных. Он также обеспечивает соблюдение особого порядка регистрации данных при выполнении обновления информационной базы, а также позволяет оптимизировать процесс регистрации данных при выполнении загрузки данных при обмене по правилам (УОП) и через универсальный формат (ОУФ).

Для каждого плана обмена создается свой набор подписок на события. В общем случае для одного плана обмена может быть создано шесть подписок на события. Параметры подписок следует задать согласно таблице, приведенной ниже. Имя подписки на событие строго регламентировано и определяется по шаблону: `<ИмяПланаОбмена><ВидПодписки>`, где `ИмяПланаОбмена` – имя плана обмена, для которого создается подписка на событие; `ВидПодписки` – вид подписки на событие (выбирается из таблицы, приведенной ниже).

Подписки на события для обмена данными:

Вид подписки	Источник	Событие
`РегистрацияДокумента`	Элементы типа `ДокументОбъект`	Перед записью
`Регистрация`	Элементы типов: `СправочникОбъект`, `ПланВидовХарактеристикОбъект`, `ПланСчетовОбъект`, `ПланВидовРасчетаОбъект`, `БизнесПроцессОбъект`, `ЗадачаОбъект`.	Перед записью
`РегистрацияНабора`	Элементы типов: `РегистрСведенийНаборЗаписей`, `РегистрНакопленияНаборЗаписей`, `РегистрБухгалтерииНаборЗаписей`.	Перед записью
`РегистрацияНабораРасчета`	Элементы типа `РегистрРасчетаНаборЗаписей`.	Перед записью
`РегистрацияКонстанты`	Элементы типа `КонстантаМенеджерЗначения`.	Перед записью
`РегистрацияУдаления`	Элементы типов: `ДокументОбъект`, `СправочникОбъект`, `ПланВидовХарактеристикОбъект`, `ПланСчетовОбъект`, `ПланВидовРасчетаОбъект`, `БизнесПроцессОбъект`, `ЗадачаОбъект`.	Перед удалением

Для назначения обработчиков событий подписок в конфигурации требуется создать общий модуль с атрибутами: **Сервер**, **Внешнее соединение**, **Клиент (обычное приложение)**, **Вызов сервера**. Имена и количество таких общих модулей не регламентированы.

В процедуре – обработчике события подписки следует прописать вызов процедуры регистрации из общего модуля `ОбменДаннымиСобытия`. Имя и параметры процедуры регистрации зависят от вида подписки. Для выбора процедуры регистрации следует воспользоваться таблицей, приведенной ниже. В качестве первого параметра процедуры регистрации следует указать имя плана обмена, все остальные параметры повторяют параметры процедуры – обработчика события подписки, из которой выполняется вызов.

Вид подписки	Процедура – обработчика события
`РегистрацияДокумента`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередЗаписьюДокумента("<ИмяПланаОбмена>", Источник, Отказ, РежимЗаписи, Реж`
`Регистрация`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередЗаписью("<ИмяПланаОбмена>", Источник, Отказ);`
`РегистрацияНабора`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередЗаписьюРегистра("<ИмяПланаОбмена>", Источник, Отказ, Замещение);`
`РегистрацияНабораРасчета`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередЗаписьюРегистра("<ИмяПланаОбмена>", Источник, Отказ, Замещение);`
`РегистрацияКонстанты`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередЗаписьюКонстанты("<ИмяПланаОбмена>", Источник, Отказ);`
`РегистрацияУдаления`	`ОбменДаннымиСобытия.МеханизмРегистрацииОбъектовПередУдалением("<ИмяПланаОбмена>", Источник, Отказ);`

Ниже перечислены объекты метаданных, которые не должны использоваться в качестве источников событий подписок:

- регистр сведений `СоответствияОбъектовИнформационныхБаз`,
- константа `НастройкиПодчиненногоУзлаРИБ`.

Настройка общих команд

Для перечисленных в таблице общих команд при первоначальном встраивании подсистемы необходимо задать свойство **Тип параметра команды**. Следует задать составной тип со ссылками на планы обмена согласно таблице 3.62.

Имя команды	ОУФ	УОП	УО	РИБ
`Синхронизировать`	+	+	+	+
`СинхронизироватьСДополнительнымиПараметрами`	+	+	–	–
`НастройкиПодключения`	+	+	+	+
`ЗагрузитьПравилаКонвертацииОбъектов`	-	+	–	–
`ЗагрузитьПравилаРегистрацииОбъектов`	+	+	+	+
`ЗагрузитьКомплектПравил`	-	+	-	-
`СценарииСинхронизации`	+	+	+	+
`СобытияОтправки`	+	+	+	+
`СобытияПолучения`	+	+	+	+
`ПолучитьНастройкиСинхронизацииДляДругойПрограммы`	+	+	+	–
`СоставОтправляемыхДанных`	+	+	+	+
`УдалитьНастройкуСинхронизации`	+	+	+	+

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Обмен данными** следует использовать роли:

№	Роли и их назначение
1.	`ВыполнениеСинхронизацииДанных`. • Интерактивный обмен данными (запуск вручную). • Мониторинг обмена данными.
2.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) • Включение и отключение подсистемы **Обмен данными**. • Добавление и изменение обменов данными. • Включение и отключение обменов данными. • Изменение префикса информационной базы. • Интерактивный обмен данными (запуск вручную). • Мониторинг обмена данными. • Удаление помеченных на удаление объектов подсистемы.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным, которые не относятся к подсистеме **Обмен данными**, но требуются для работы с ней.

№	Вспомогательные роли и их назначение
1.	`ЧтениеУзловПлановОбмена`. Роль для чтения узлов планов обмена конфигурации.

Примеры настройки прав доступа пользователей при работе в локальном режиме приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор обмена: • включение/отключение подсистемы в конфигурации; • удаление помеченных элементов подсистемы; • назначение префикса узла распределенной информационной базы; • создание новых обменов данными; • изменение существующих обменов данными; • выполнение обмена; • мониторинг обмена.	• `ПолныеПрава` (из подсистемы **Базовая функциональность**), • `АдминистраторСистемы` (из подсистемы **Базовая функциональность**)
2.	Пользователь обмена: • выполнение обмена; • мониторинг обмена.	• `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ПросмотрЖурналаРегистрации` (из подсистемы **Базовая функциональность**), • `ВыполнениеСинхронизацииДанных`, • `ЧтениеУзловПлановОбмена`.

Также следует учитывать, что при создании и выполнении обмена данными через внешнее соединение в базе-корреспонденте должен использоваться пользователь с ролями `АдминистраторСистемы` и `ПолныеПрава`.

Настройка прав доступа пользователей при внедрении подсистемы «Версионирование объектов»

При внедрении подсистемы **Версионирование объектов** она используется для регистрации конфликтов и запретов загрузки данных по установленной дате. Для их просмотра необходимо назначить роль `ЧтениеИнформацииОВерсияхОбъектов`.

Настройка прав доступа пользователей при работе в модели сервиса

При работе конфигурации в модели сервиса пользователям областей данных необходимо назначить роль `ВыполнениеСинхронизацииДанных`. Настройка обменов данными выполняется администратором абонента в контексте конфигурации **Менеджер сервиса**.

Использование при разработке конфигурации

В процессе обмена данными при чтении файла сообщения обмена важно не выполнять дополнительных проверок на заполнение реквизитов объектов. Для этого в обработчиках событий `ПередЗаписью`, `ПриЗаписи` и `ПередУдалением` для каждого объекта метаданных, который участвует в обмене, следует использовать конструкцию:

```
Если ОбменДанными.Загрузка Тогда
   Возврат;
КонецЕсли;
```

Этот код следует располагать в самом начале обработчиков `ПередЗаписью`, `ПриЗаписи` и `ПередУдалением` в модуле объекта, а также в подписках на события этих типов.

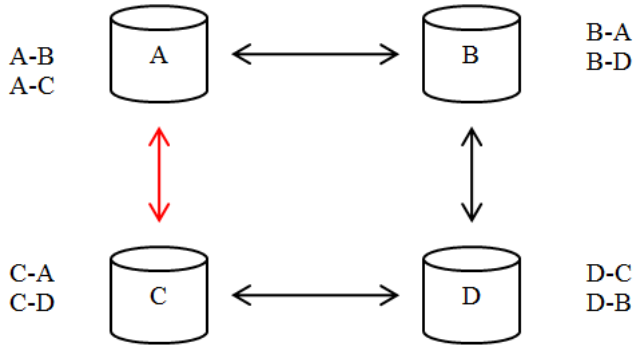
Настройка обмена данными

Объекты метаданных подсистемы должны быть включены в состав планов обмена в соответствии с описанием в разделе Состав плана обмена.

Общие сведения для всех видов обмена данными

Топология обмена данными

Подсистема не накладывает ограничений на топологию обмена и на порядок настройки обмена в гетерогенных системах. Следует создавать обмены таким образом, чтобы избежать возникновения волн миграции данных. Волны миграции данных – это явление, при котором наблюдается бесконечная миграция изменений данных между информационными базами, участвующими в обмене. Они возникают в случае неправильной настройки топологии обмена (в разрезе информационных баз, планов обмена, объектов метаданных). Волны миграции данных имеют тенденцию к постоянному увеличению при работе пользователей в обменивающихся информационных базах.



Топология обмена, при которой возникают волны миграции данных

На рисунке показаны четыре ИБ, между которыми настроен обмен. Каждая ИБ имеет свой уникальный префикс (код): A, B, C и D. Благодаря этому может быть настроена произвольная топология обмена. На рисунке показаны сочетания пар предопределенного узла и узла ИБ-корреспондента для каждой ИБ: A–B, A–C и др. Красным цветом показан обмен, который может спровоцировать возникновение волн миграции данных. При создании топологии обмена следует избегать таких обменов.

Использование значений по умолчанию в обмене данными

В процессе обмена данными может возникнуть необходимость дополнить данные, полученные от корреспондента. Например, документ не может быть проведен, пока в нем не указана валюта взаиморасчетов. В исходных данных, полученных от корреспондента, валюта не предусмотрена. Поэтому значение валюты необходимо подставить в документ на этапе загрузки данных. Это значение можно, например, запомнить в свойствах узла плана обмена. На этапе настройки обмена можно будет задать необходимые значения. В случае обмена по правилам использование значений по умолчанию производится в обработчиках правил обмена – например, в обработчике правила конвертации объекта **После загрузки**.

Желательно проектировать обмен данными таким образом, чтобы при настройке обмена от пользователя не требовалось указывать эти значения. Все необходимые настройки должны устанавливаться системой или вообще отсутствовать. Только в исключительных случаях допускается выводить настройки значений по умолчанию на пользователя.

Коды узлов плана обмена

В локальном режиме работы коды узлов планов обмена совпадают с префиксами информационных баз, участвующих в обмене. Предопределенному узлу плана обмена назначается код, равный префиксу текущей информационной базы. Коды узлов назначаются системой в момент создания новой настройки обмена данными.

В модели сервиса коды узлов планов обмена соответствуют номерам областей данных, которым они принадлежат, с префиксом «S». Префикс «S» используется для разделения множества кодов узлов, работающих в модели сервиса, и множества кодов узлов, работающих в локальном режиме. Предопределенному узлу плана обмена назначается код, равный номеру текущей области данных, с префиксом «S». Код предопределенного узла назначается системой в момент создания области данных.

Исключение: для узлов плана обмена ОУФ начиная с версии формата 1.5 в качестве кода назначаются уникальные идентификаторы, не связанные с префиксом информационной базы и с номером области.

Настройка каталогов обмена данными для работы в модели сервиса

Для информационной базы, которая работает в модели сервиса, необходимо задать значение константы `КаталогСообщенийОбменаДаннымиДляWindows` или значение константы `КаталогСообщенийОбменаДаннымиДляLinux`, если сервер **1С:Предприятия** работает под управлением ОС MS Windows или ОС Linux соответственно. Если кластер сервера **1С:Предприятия** развернут на нескольких машинах с разными ОС, то необходимо задать значение обеих констант. В этом случае значения констант должны указывать на один и тот же сетевой каталог.

> **Важно!**
>
> Если две информационные базы, между которыми настроен обмен данными, расположены в одной локальной сети или на одной машине, то рекомендуется, чтобы константы `КаталогСообщенийОбменаДаннымиДляWindows` и `КаталогСообщенийОбменаДаннымиДляLinux` в обеих базах указывали на один и тот же каталог обмена. Это позволит повысить производительность обмена данными.

Если кластер сервера **1С:Предприятия** работает на одном физическом сервере, то задание значений констант не является обязательным. В качестве каталога временных файлов для сообщений обмена будет использоваться временный каталог пользователя, от имени которого запущен сервер **1С:Предприятия**.

Обновление правил обмена

В процессе разработки конфигурации в нее могут быть внесены изменения, которые потребуют изменения правил обмена (только для `УОП`) и правил регистрации. Кроме того, правила обмена и правила регистрации могут быть изменены ввиду исправления ошибок в самих правилах. Правила следует также обновлять при изменении внутреннего формата правил. Для обновления правил необходимо выполнить обновление соответствующих макетов правил и обработок в конфигураторе. При изменении версии конфигурации обновление правил будет выполнено автоматически подсистемой обновления версии ИБ. Обновлению подлежат только типовые правила, загруженные из макета конфигурации. Нетиповые правила, загруженные из внешнего файла, не обновляются.

> **Важно!**
>
> Правила обмена загружаются в информационную базу. Поэтому при разработке правил обмена без изменения версии конфигурации автоматического обновления правил не произойдет. В таком случае необходимо выполнять обновление правил вручную. Для этого следует зайти в форму загрузки правил (команда **Открыть правила конвертации объектов** или **Открыть правила регистрации объектов**) и перечитать правила из макета конфигурации или из внешнего файла.

Сценарий выгрузки объектов «При необходимости»

Для универсального обмена данными по правилам обмена и для обмена через универсальный формат подсистема позволяет использовать сценарий обмена **При необходимости**. Суть сценария заключается в том, что объекты выгружаются не всегда, а только в том случае, если на объект имеются ссылки из ранее выгруженных объектов. Например, справочник **Номенклатура** может содержать значительное количество элементов, при этом обмениваться необходимо не всеми элементами, а только теми, которые выгружаются вместе с документами прихода/расхода номенклатуры. Если элемент номенклатуры был выгружен хотя бы один раз, то изменения элемента постоянно синхронизируются между обменивающимися конфигурациями.

Для использования сценария на примере справочника номенклатуры в системе необходимо:

- создать в плане обмена отдельный реквизит шапки – переключатель режима выгрузки с именем `РежимВыгрузкиНоменклатуры`, тип `ПеречислениеСсылка.РежимыВыгрузкиОбъектовОбмена`;
- значение реквизита `РежимВыгрузкиНоменклатуры` установить в значение `Перечисления.РежимыВыгрузкиОбъектовОбмена.ВыгружатьПриНеобходимости`;
- создать правило регистрации объекта (ПРО) для справочника номенклатуры, в котором дополнительно задать значение поля переключателя режима равным `РежимВыгрузкиНоменклатуры`.

Для одного переключателя режима выгрузки `РежимВыгрузкиНоменклатуры` можно использовать несколько объектов метаданных. Например, при выгрузке номенклатуры также можно выгружать единицы измерения номенклатуры. Единицы измерения номенклатуры в этом случае будут тоже выгружаться только при необходимости.

Обработчик события «Регистрация изменений начальной выгрузки данных»

Для универсального обмена данными с использованием правил обмена, обмена через универсальный формат и для универсального обмена данными без правил обмена предусмотрен обработчик события **Регистрация изменений начальной выгрузки данных**. Он не используется для обменов в РИБ.

Обработчик располагается в общем модуле `ОбменДаннымиПереопределяемый` и используется для переопределения стандартной обработки регистрации изменений начальной выгрузки данных. При стандартной обработке будут зарегистрированы изменения всех данных из состава плана обмена. Если для плана обмена предусмотрены фильтры ограничения миграции данных, то использование этого обработчика позволит повысить производительность начальной выгрузки данных (в некоторых случаях в среднем в 2–4 раза по сравнению со стандартной обработкой).

В обработчике можно реализовать регистрацию изменений с учетом фильтров ограничения миграции данных. Если для плана обмена используются ограничения миграции по дате или по дате и организациям, то можно воспользоваться универсальной процедурой `ОбменДаннымиСевер.ЗарегистрироватьДанныеПоДатеНачалаВыгрузкиИОрганизациям`. Пример реализации обработчика можно посмотреть в демонстрационной конфигурации.

Обработчик события «При выгрузке данных» для планов обмена ОУФ, УОП и УО

Обработчик располагается в общем модуле `ОбменДаннымиПереопределяемый` и используется для переопределения стандартной процедуры выгрузки данных в файл сообщения обмена. При стандартной обработке данные будут выгружены в файл в соответствии с видом плана обмена – с использованием правил конвертации или с использованием платформенной XML-сериализации данных. В данном обработчике может быть реализована произвольная логика выгрузки данных: выборка данных для выгрузки, сериализация данных в файл сообщения или сериализация данных в поток. После выполнения обработчика выгруженные данные будут отправлены получателю подсистемой обмена данными. Формат сообщения для выгрузки может быть произвольным.

Обработчик события «При загрузке данных» для планов обмена ОУФ, УОП и УО

Обработчик располагается в общем модуле `ОбменДаннымиПереопределяемый` и используется для переопределения стандартной процедуры загрузки данных из файла сообщения обмена. При стандартной обработке данные будут загружены из файла в соответствии с видом плана обмена – с использованием правил конвертации или с использованием платформенной XML-сериализации данных. В данном обработчике может быть реализована произвольная логика загрузки данных: необходимые проверки перед загрузкой данных, сериализация данных из файла сообщения или сериализация данных из потока. Формат сообщения для загрузки может быть произвольным.

Обработчик события «При получении доступных версий формата»

Обработчик располагается в общем модуле `ОбменДаннымиПереопределяемый` и используется для переопределения стандартной процедуры получения поддерживаемых в конфигурации версий формата обмена `EnterpriseData` для использования в обработке `ВыгрузкаЗагрузкаEnterpriseData`.

Обработчик события «При получении доступных расширений формата»

Обработчик располагается в общем модуле `ОбменДаннымиПереопределяемый` и используется для переопределения стандартной процедуры получения поддерживаемых в конфигурации схем расширений формата обмена `EnterpriseData` для использования в обработке `ВыгрузкаЗагрузкаEnterpriseData`.

Общие данные узлов

Для планов обмена двух обменивающихся информационных баз могут быть выделены так называемые общие данные узлов. Общие данные узлов – это реквизиты, которые присутствуют в планах обмена обеих конфигураций и значения которых должны быть одинаковыми в узлах обмена информационных баз-корреспондентов.

В качестве примера использования механизма общих данных узлов можно рассмотреть задачу разработки обмена с настройкой **Дата начала выгрузки документов**. Если в обеих конфигурациях в модуле менеджера плана обмена реквизит, управляющий данной настройкой, будет объявлен как общие данные узлов, то автоматически будет поддерживаться синхронность данной настройки в информационных базах-корреспондентах.

Не рекомендуется включать в общие данные узлов реквизиты ссылочного типа, если объекты данных, соответствующие этим типам, могут иметь разные уникальные идентификаторы в информационных базах, участвующих в обмене.

Коллизии изменения данных

При настроенном обмене между двумя базами возникают ситуации, когда одни и те же данные изменяются одновременно (в промежутке между выполнением сеанса обмена данными) в обеих информационных базах. Это приводит к возникновению двух различающихся версий одинаковых данных в обменивающихся информационных базах. Такая ситуация называется коллизией (конфликтом) изменения данных.

При возникновении коллизии изменения данных в информационную базу будет загружена та версия, которая имеет более высокий приоритет. При одинаковом приоритете обеих версий приоритет будет назначен автоматически, исходя из значений по умолчанию в зависимости от вида обмена данными:

- РИБ – изменения данных в главном узле имеют приоритет по отношению к изменениям данных в подчиненном узле;
- ОУФ, УО, УОП – приоритет имеют данные, загружаемые из другой информационной базы.

Разработчик имеет возможность переопределить приоритеты для возникающих коллизий при помощи процедуры `ОбменДаннымиПереопределяемый.ПриКоллизииИзмененийДанных` (подробнее см. комментарий к процедуре).

Следует учитывать, что для однозначного определения разрешения коллизии необходимо устанавливать приоритеты при возникновении коллизий изменения данных таким образом, чтобы они были противоположными для двух обменивающихся информационных баз. То есть если в первой информационной базе в случае коллизии данные принимаются, то во второй информационной базе при коллизии они должны быть отклонены.

РИБ

Особенности создания начального образа подчиненного узла распределенной ИБ

Если объект метаданных нужно использовать только на момент создания начального образа подчиненного узла в распределенной ИБ, то этот объект метаданных требуется включить в состав плана обмена с отключенным признаком авторегистрации. Также необходимо исключить его из состава всех подписок на события подсистемы **Обмен данными**.

Например, справочник номенклатуры не должен синхронизироваться между узлами РИБ. В каждом узле справочник номенклатуры специфичен и ведется независимо от других узлов. Однако вновь создаваемый подчиненный узел должен содержать заполненный по умолчанию справочник номенклатуры (обязательная структура папок и элементов).

Обмен через универсальный формат

Спецификация сообщения обмена

URI пространства имен: http://www.1c.ru/SSL/Exchange/Message

Элемент	Тип	Формат	Описание
`<message></message>`	• M		• Корневой элемент сообщения обмена
`<header></header>`	• M		• Заголовочный элемент
`<format></format>`	• M	• string	• Формат сообщения обмена. URI пространства имен соответствующего пакета EnterpriseData
`<creationdate></creationdate>`	• M	• dateTime	• Дата создания сообщения обмена
`<confirmation></confirmation>`	• O		• Имеет смысл только при обмене с использованием планов обмена. Содержит информацию о плане обмена и информацию для квитирования
`<exchangeplan></exchangeplan>`	• M	• string	• Имя плана обмена
`<to></to>`	• M	• string	• Идентификатор получателя сообщения обмена. См. Коды узлов плана обмена
`<from></from>`	• M	• string	• Идентификатор отправителя сообщения обмена. См. Коды узлов плана обмена
`<messageno></messageno>`	• M	• integer	• Номер текущего сообщения обмена в системе отправителя
`<receivedno></receivedno>`	• M	• integer	• Номер последнего сообщения обмена, принятого отправителем
	• -		• -
`<availableversion></availableversion>`	• M[1..100]	• string	• Перечень поддерживаемых версий формата в виде номера версии, например **1.6**.
`<newfrom></newfrom>`	• O	• string	• Идентификатор отправителя при переходе от кодирования узлов префиксами на кодирование уникальными идентификаторами. Зарезервирован для служебного использования
`<availableobjecttypes></availableobjecttypes>`	• O		• Корневой элемент для перечня поддерживаемых объектов формата
`<objecttype></objecttype>`	• O[0..n]		• Набор элементов, описывающих поддержку объектов формата на отправку и получение в разрезе версий
`<name></name>`	• M	• string	• Имя объекта формата, например `Справочник.Контрагенты`.
`<sending></sending>`	• M	• string	• Список версий через «,», для которых поддерживается отправка объекта. Например, «1.4,1.5,1.6». Если список версий соответствует всем поддерживаемым версиям, описанным в последовательности AvailableVersion, то указывается символ «*».
`<receiving></receiving>`	• M	• string	• Список версий через «,», для которых поддерживается получение объекта. Например, «1.4,1.5,1.6». Если список версий соответствует всем поддерживаемым версиям, описанным в последовательности AvailableVersion, то указывается символ «*».
	• -		• -

Элемент	Тип	Формат	Описание
	• -		• -
`<prefix></prefix>`	• O	• string	• Префикс информационной базы – отправителя сообщения. Используется при переходе на кодирование узлов обмена уникальными идентификаторами. Зарезервирован для служебного использования
	• -		• -
	• M		• Элемент, содержащий объекты формата с данными
	• -		

Используемые обозначения:

- **Тип**: M – обязательный элемент; O – необязательный элемент; M[1..K] – последовательность элементов в количестве от 1 до K, где K – целое неотрицательное число >= 1; O[0..n] – последовательность элементов в количестве от 0 до n, где n – целое неотрицательное число >= 0. Если в качестве n не указано конкретное значение, количество элементов в последовательности не ограничено.
- **Формат**: тип значения элемента из множества типов, описанных в XML-схеме http://www.w3.org/2001/XMLSchema; если не указан, то один из типов, описанных в схеме www.1c.ru/SSL/Exchange/Message.

Настройка обмена в режиме пассивного подключения

При создании новой синхронизации данных через Интернет в пассивном режиме потребуется указать файл с настройками XDTO корреспондента. Спецификация файла соответствует спецификации файла сообщения обмена, со следующими исключениями:

В `Header.Format` должен быть указан базовый URI пакетов EnterpriseData без версии, т. е. http://v8.1c.ru/edi/edi_stnd/EnterpriseData.

- В `Header.Confirmation.MessageNo` и `Header.Confirmation.ReceivedNo` должны быть указаны «0».
- Элемент `Body` должен отсутствовать.

Пример файла настроек, используемого для настройки синхронизации данных в пассивном режиме.

```
<?xml version="1.0" encoding="UTF-8"?>
<Message xmlns:msg="http://www.1c.ru/SSL/Exchange/Message" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSch
    <msg:Header>
        <msg:Format>http://v8.1c.ru/edi/edi_stnd/EnterpriseData</msg:Format>
        <msg:CreationDate>2018-06-04T14:01:18</msg:CreationDate>
        <msg:Confirmation>
            <msg:ExchangePlan>СинхронизацияДанныхЧерезУниверсальныйФормат</msg:ExchangePlan>
            <msg:To>31</msg:To>
            <msg:From>30</msg:From>
            <msg:MessageNo>0</msg:MessageNo>
            <msg:ReceivedNo>0</msg:ReceivedNo>
        </msg:Confirmation>
        <msg:AvailableVersion>1.6</msg:AvailableVersion>
        <msg:AvailableVersion>1.5</msg:AvailableVersion>
        <msg:AvailableVersion>1.4</msg:AvailableVersion>
        <msg:AvailableVersion>1.3</msg:AvailableVersion>
        <msg:AvailableVersion>1.2</msg:AvailableVersion>
        <msg:NewFrom>1a8ac3e6-ec58-484f-9257-1e77171ce21d</msg:NewFrom>
        <msg:AvailableObjectTypes>
            <msg:ObjectType>
                <msg:Name>Документ.РеализацияТоваровУслуг</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving>*</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Документ.СписаниеЗапасов</msg:Name>
                <msg:Sending>1.3,1.4,1.5,1.6</msg:Sending>
                <msg:Receiving>1.3,1.4,1.5,1.6</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Документ.СчетПокупателю</msg:Name>
                <msg:Sending>1.3,1.4,1.5,1.6</msg:Sending>
                <msg:Receiving>1.3,1.4,1.5,1.6</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.БанковскиеСчета</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving>*</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.ЕдиницыИзмерения</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving/>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.Контрагенты</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving>*</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.Номенклатура</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving>*</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.Организации</msg:Name>
                <msg:Sending>*</msg:Sending>
                <msg:Receiving>*</msg:Receiving>
            </msg:ObjectType>
            <msg:ObjectType>
                <msg:Name>Справочник.ОтветственныеЛица</msg:Name>
                <msg:Sending>1.3,1.4,1.5,1.6</msg:Sending>
                <msg:Receiving>1.3,1.4,1.5,1.6</msg:Receiving>
            </msg:ObjectType>
        </msg:AvailableObjectTypes>
        <msg:Prefix>30</msg:Prefix>
    </msg:Header>
</Message>
```

Модуль менеджера обмена через универсальный формат

Процедуры и функции, полностью описывающие правила выгрузки данных из информационной базы в формат обмена и правила загрузки данных из формата обмена в информационную базу, разрабатываются в общем модуле – модуле менеджера обмена через универсальный формат.

Модуль создается автоматически с помощью конфигурации **Конвертация данных**, редакция 3.0, на основе настроенных правил обмена либо вручную в конфигураторе.

При описании структуры модуля используются следующие сокращения:

- `ПОД` – правило обработки данных;
- `ПКО` – правило конвертации объекта;
- `ПКПД` – правило конвертации предопределенных данных;
- `ПКС` – правило конвертации свойства.

Модуль состоит из нескольких крупных разделов, каждый из которых содержит свою группу процедур и функций.

Комментарий. Первая строка модуля содержит комментарий с наименованием конвертации. Эта строка необходима для идентификации модуля при использовании команды **Загрузка обработчиков** в программе **Конвертация данных**, редакция 3.0.

Процедуры конвертации. Содержит предопределенные процедуры, которые выполняются на разных этапах синхронизации данных: перед конвертацией, после конвертации, перед отложенным заполнением.

Правила обработки данных (ПОД). Содержит процедуры и функции, которые описывают правила обработки данных.

Правила конвертации объектов (ПКО). Содержит процедуры и функции, которые описывают правила конвертации объектов, а также правила конвертации свойств данных объектов.

Правила конвертации предопределенных данных (ПКПД). Содержит процедуру, заполняющую правила конвертации предопределенных данных.

Алгоритмы. Содержит произвольные алгоритмы, которые вызываются из других правил (ПОД или ПКО).

Параметры. Содержит логику заполнения параметров конвертации.

Общего назначения. Содержит процедуры и функции, которые широко используются в правилах и алгоритмах.

Ниже описаны параметры процедур и функций, которые используются в нескольких видах процедур модуля менеджера.

`КомпонентыОбмена`. Тип – `Структура`. Содержит параметры и правила обмена, инициализированные в рамках выполнения сеанса обмена.

`НаправлениеОбмена`. Тип – `Строка`. Либо `Отправка`, либо `Получение`.

`ДанныеИБ`. Тип – `СправочникОбъект`. Либо `ДокументОбъект`.

Процедуры, связанные с событиями конвертации

Предусмотрены три предопределенные процедуры, которые вызываются в процессе конвертации:

- `ПередКонвертацией`. Вызывается перед выполнением синхронизации данных. Обычно в этой процедуре размещается логика инициализации различных параметров конвертации, заполнения значений по умолчанию и т. д. Параметры: `КомпонентыОбмена`.
- `ПослеКонвертации`. Вызывается после выполнения синхронизации данных, но до выполнения отложенного заполнения. Параметры: `КомпонентыОбмена`.
- `ПередОтложеннымЗаполнением`. Вызывается перед выполнением отложенного заполнения. Здесь может быть расположена логика сортировки или корректировки таблицы объектов, подлежащих отложенному заполнению. Параметры: `КомпонентыОбмена`.

Процедуры ПОД

`ЗаполнитьПравилаОбработкиДанных`. Экспортная процедура, в которой располагается логика заполнения правил обработки данных. Содержит вызовы других процедур, которые добавляют в таблицу правил правило обработки конкретного объекта (см. ниже процедуры `ДобавитьПОД`). Параметры: `НаправлениеОбмена`, `ПравилаОбработкиДанных` (таблица значений, инициализированная в рамках выполнения сеанса обмена).

`ДобавитьПОД_<ИмяПОД>`. Набор процедур, которые наполняют таблицу ПОД правилами для конкретных объектов. Количество таких процедур соответствует количеству ПОД, предусмотренных для данной конвертации в программе **Конвертация данных**, редакция 3.0. Параметры: `ПравилаОбработкиДанных` (таблица значений, инициализированная в рамках выполнения сеанса обмена).

`ПОД_<ИмяПОД>_ПриОбработке`. Процедура содержит текст обработчика `ПриОбработке` для конкретного ПОД. Обработчик предназначен для реализации логики конвертации на уровне объектов. Например, назначить конкретному объекту определенное ПКО в зависимости от содержимого объекта. Параметры:

- `ДанныеИБ` либо `ДанныеXDTO` (в зависимости от направления обмена):
  - при отправке – объект (`СправочникОбъект`, `ДокументОбъект`);
  - при получении – структуру с описанием объекта XDTO.
- `ИспользованиеПКО`. Тип – `Структура`. Ключ содержит строку с именем ПКО, а значение типа `Булево` (`Истина` – ПКО используется, `Ложь` – ПКО не используется).
- `КомпонентыОбмена`.

`ПОД_<ИмяПОД>_ВыборкаДанных`. Функция содержит текст обработчика `ПриВыгрузке`. Обработчик предназначен для реализации произвольного алгоритма выборки объектов, подлежащих выгрузке. Возвращаемое значение: массив объектов, подлежащих выгрузке. В массиве могут содержаться как ссылки на объекты информационной базы, так и структура с данными для выгрузки. Параметры: `КомпонентыОбмена`.

Процедуры ПКО

`ЗаполнитьПравилаКонвертацииОбъектов`. Экспортная процедура, в которой располагается логика заполнения правил конвертации объектов. Содержит вызовы других процедур, которые добавляют в таблицу правил правило конвертации конкретного объекта (см. ниже процедуры `ДобавитьПКО`). Параметры: `НаправлениеОбмена`, `ПравилаКонвертации` (таблица значений, инициализированная в рамках выполнения сеанса обмена).

`ДобавитьПКО_<ИмяПКО>`. Набор процедур, которые наполняют таблицу ПКО правилами для конкретных объектов. Количество таких процедур соответствует количеству ПКО, предусмотренных для данной конвертации в программе **Конвертация данных**, редакция 3.0. Параметры: `ПравилаКонвертации` (таблица значений, инициализированная в рамках выполнения сеанса обмена).

`ПКО_<ИмяПКО>_ПриОтправкеДанных`. Процедура содержит текст обработчика `ПриОтправке` для конкретного ПКО. Обработчик используется при выгрузке данных. Предназначен для реализации логики конвертации данных, содержащихся в объекте информационной базы, в описание объекта XDTO. Параметры:

- `ДанныеИБ`. Тип – `СправочникОбъект`, `ДокументОбъект`. Обрабатываемый объект информационной базы.
- `ДанныеXDTO`. Тип – `Структура`. Предназначен для доступа к данным объекта XDTO.
- `КомпонентыОбмена`.
- `СтекВыгрузки`. Тип – `Массив`. Содержит ссылки на выгружаемые объекты с учетом вложенности.

В обработчике возможно отказаться от выполнения конвертации данных. Для этого параметру `ДанныеXDTO` следует установить значение `Неопределено`.

`ПКО_<ИмяПКО>_ПриКонвертацииДанныхXDTO`. Процедура содержит текст обработчика `ПриКонвертацииДанныхXDTO` для конкретного ПКО. Обработчик используется при загрузке данных. Предназначен для реализации произвольной логики конвертации данных XDTO. Параметры:

- `ДанныеXDTO`. Тип – `Структура`. Свойства объекта XDTO, прошедшие предварительную обработку для упрощения доступа к ним.
- `ПолученныеДанные`. Тип – `СправочникОбъект`, `ДокументОбъект`. Объект информационной базы, сформированный путем конвертации данных XDTO. Не записан в информационную базу.
- `КомпонентыОбмена`.

В данном обработчике описываются инструкции конвертации свойств, для которых указан признак **Используется алгоритм конвертации**. Инструкция представляет собой структуру с ключами:

- `ИмяПКО`. Строковый идентификатор правила конвертации.
- `Значение`. Структура ключевых свойств, полученная из `ДанныеXDTO`, которая должна быть сконвертирована в реквизит объекта данных.
- `УдалятьСозданныеПоКлючевымСвойствам`. Признак необходимости удаления созданного по ключевым свойствам объекта после загрузки всех данных. Удалены будут объекты, загруженные «по ссылке», т.е. которые присутствуют только в составе свойств других объектов формата. По умолчанию такие объекты не удаляются. Если объект был загружен «по ссылке» и как самостоятельный объект, то он не будет удален вне зависимости от значения данного ключа.

Для указания необходимости конвертации реквизита по инструкции следует добавить ее в дополнительные свойства объекта `ПолученныеДанные` с ключом, соответствующим имени конвертируемого реквизита.

```
ИнструкцияКонтрагент = Новый Структура("ИмяПКО, Значение", "Справочник_Контрагенты_Получение", ДанныеXDTO.КлючевыеСвойства.Контрагент);
ПолученныеДанные.ДополнительныеСвойства.Вставить("Контрагент", ИнструкцияКонтрагент);
```

`ПКО_<ИмяПКО>_ПередЗаписьюПолученныхДанных`. Процедура содержит текст обработчика `ПередЗаписьюПолученныхДанных` для конкретного ПКО. Обработчик используется при загрузке данных. Предназначена для реализации дополнительной логики, которую необходимо выполнить перед записью объекта в информационную базу. Например, нужно ли загрузить изменения в существующие данные ИБ либо следует загрузить их как новые данные. Параметры:

- `ПолученныеДанные`. Тип – `СправочникОбъект`, `ДокументОбъект`. Элемент данных, сформированный путем конвертации данных XDTO. Записывается в случае, если эти данные являются для информационной базы новыми (параметр `ДанныеИБ` содержит значение `Неопределено`). В противном случае `ПолученныеДанные` замещают собой `ДанныеИБ` (все свойства из `ПолученныеДанные` переносятся в `ДанныеИБ`). Если стандартное замещение данных ИБ полученными данными не требуется, следует прописать свою логику переноса, после чего установить параметру `ПолученныеДанные` значение `Неопределено`.
- `ДанныеИБ`. Тип – `СправочникОбъект`, `ДокументОбъект`. Элемент данных информационной базы, соответствующий полученным данным. Если соответствующие данные не найдены, содержит `Неопределено`.
- `КонвертацияСвойств`. Тип – `Таблица значений`. Содержит правила конвертации свойств текущего объекта, инициализированные в рамках выполнения сеанса обмена.
- `КомпонентыОбмена`.

В обработчике возможно отказаться от записи данных. Для этого параметрам `ПолученныеДанные` и `ДанныеИБ` следует установить значение `Неопределено`.

Процедуры ПКПД

`ЗаполнитьПравилаКонвертацииПредопределенныхДанных`. Экспортная процедура, в которой располагается логика заполнения правил конвертации предопределенных данных. Параметры: `НаправлениеОбмена`, `ПравилаКонвертации` (таблица значений, инициализированная в рамках выполнения сеанса обмена).

Алгоритмы

В программе **Конвертация данных**, редакция 3.0 есть возможность создавать произвольные алгоритмы, которые вызываются из обработчиков ПОД и ПКПД. Наименование, параметры и содержимое алгоритмов определяются при разработке правил.

Параметры

`ЗаполнитьПараметрыКонвертации`. Экспортная процедура, в которой происходит заполнение структуры с параметрами конвертации. Параметры: `ПараметрыКонвертации` (тип – `Структура`).

Процедуры и функции общего назначения

`ВыполнитьПроцедуруМодуляМенеджера`. Параметры: `ИмяПроцедуры` (строка), `Параметры` (структура). Экспортная процедура, которая предназначена для вызова неэкспортной процедуры модуля, имя и параметры которой получены на вход. Позволяет выполнить вызов процедуры или функции по строке без использования метода `Выполнить`.

`ВыполнитьФункциюМодуляМенеджера`. Параметры: `ИмяПроцедуры` (строка), `Параметры` (структура). Функция, назначение аналогично `ВыполнитьПроцедуруМодуляМенеджера`. Отличие в том, что она вызывает функцию и возвращает ее значение.

Универсальный обмен по правилам

Безопасное выполнение кода обработчиков при использовании обработки «УниверсальныйОбменДаннымиXML»

Для конфигураций, обменивающихся при помощи обработки `УниверсальныйОбменДаннымиXML` (УОД), можно повысить уровень безопасности путем выполнения кода обработчиков загрузки из обработки в составе конфигурации. Для того чтобы сгенерировать модуль с кодом обработчиков загрузки, необходимо:

1. Запустить базу-источник и сформировать файл выгрузки на основании правил обмена.
2. Запустить базу-приемник, запустить в ней УОД и на основании файла выгрузки сформировать текст модуля обработчиков загрузки.
3. Скопировать полученный текст модуля в обработку в составе конфигурации.
4. Подключить полученную обработку при помощи конструкции вида:

```
УОД = Обработки.УниверсальныйОбменДаннымиXML.Создать();
УОД.ФлагРежимОтладкиОбработчиков = Истина;
УОД.ИмяФайлаВнешнейОбработкиОбработчиковСобытий = "<ИмяОбработки>";
УОД.ВыполнитьЗагрузку();
```

Режим отладки

Режим отладки позволяет разрабатывать и выполнять отладку кода обработчиков с использованием конфигуратора в случае обмена по правилам конвертации. В этом режиме код обработчиков выгрузки и/или загрузки выполняется из внешних обработок, что позволяет вносить в него изменения без перезапуска конфигурации. Для использования данного режима необходимо:

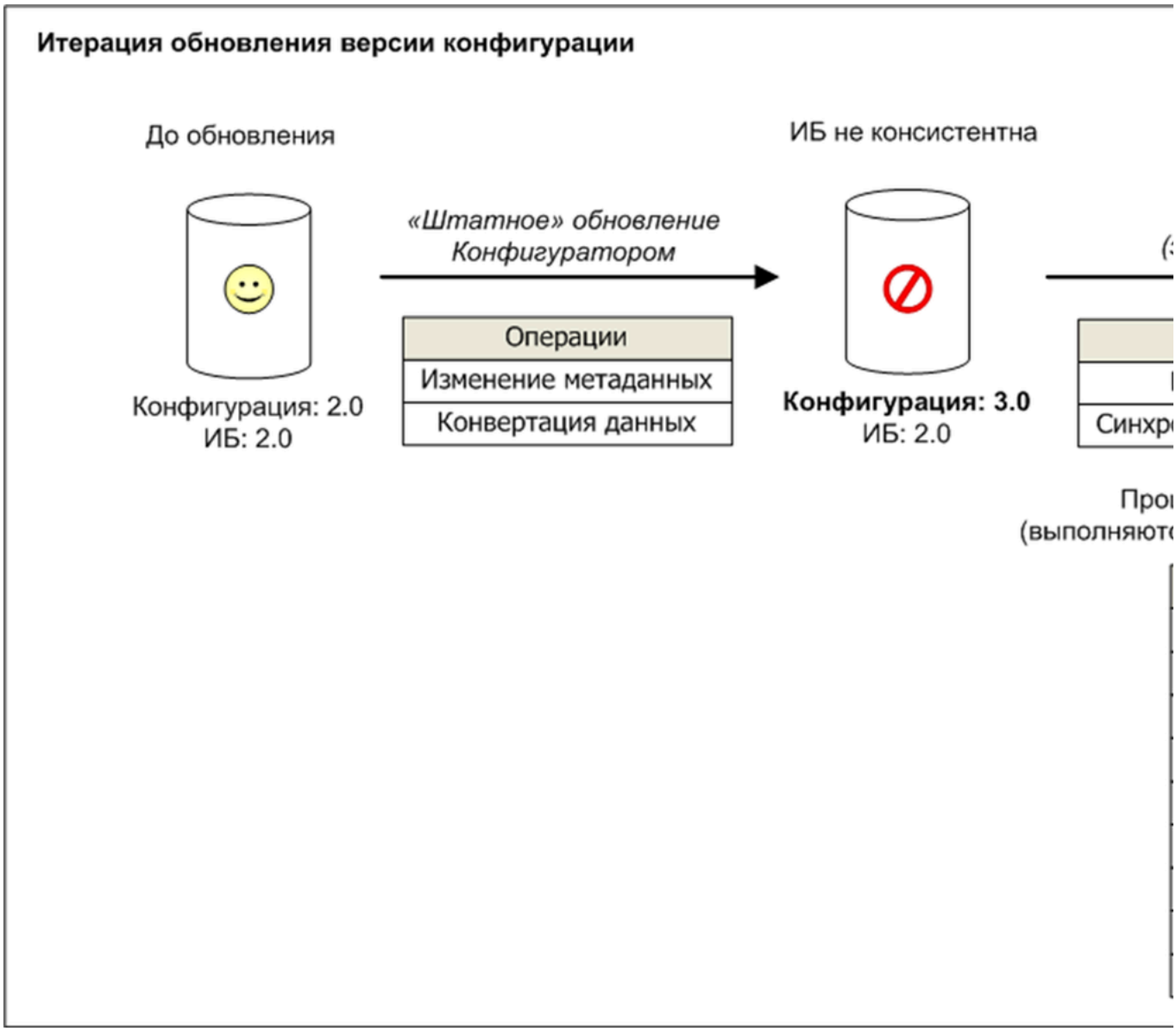
- В инструменте **Конвертация данных** версии 2.1.6 и выше:
  - Отключить режим совместимости в свойствах конвертации.
  - Сохранить правила конвертации объектов.
  - Сформировать отладочные модули, содержащие код обработчиков, и скопировать их в модуль внешней обработки.
- В конфигурации:
  - Загрузить правила конвертации, сохраненные в КД.
  - В настройках правил конвертации включить режим отладки.
  - Подключить внешние обработки, сформированные в КД.
  - Запустить обмен данными.
  - При этом в режиме **Конфигуратор** можно открыть подключенную внешнюю обработку, отлаживать и дорабатывать ее код.
- Имеется возможность перенести все сделанные в обработке изменения в правила обмена, используя инструмент **Конвертация данных** версии 2.1.6 и выше.

Режим отладки недоступен при работе в модели сервиса по соображениям безопасности.

Обновление версии ИБ

Подсистема **Обновление версии ИБ** предоставляет программный интерфейс для выполнения процедур-обработчиков по первоначальному заполнению и обновлению данных информационной базы (ИБ) при изменении версии конфигурации, а также позволяет выводить отчет об изменениях в новой версии конфигурации.

Принципиальная схема обновления конфигурации информационной базы представлена на рисунке.



При запуске клиентского приложения подсистема **Обновление версии ИБ** проверяет, не изменилась ли конфигурация. Если версия конфигурации отличается от версии, сохраненной в информационной базе, то при запуске системы с административными правами выполняется обновление информационной базы. В этом случае подсистема **Обновление версии ИБ** последовательно выполняет процедуры – обработчики обновления в интервале от номера версии информационной базы до номера версии конфигурации включительно и записывает в информационную базу текущую версию конфигурации.

После обновления информационной базы подсистема **Обновление версии ИБ** выводит форму **Описание обновлений**, в которой администратор может ознакомиться с описанием изменений в текущей версии конфигурации.

Если у пользователя недостаточно прав (нет прав `МонопольныйРежим` и `Администрирование`) или при обновлении происходит ошибка, то запуск системы останавливается.

Если в конфигурации предусмотрены отложенные процедуры – обработчики обновления, то в клиент-серверной версии они запускаются в фоне с помощью регламентного задания **Отложенное обновление ИБ** параллельно с началом работы пользователей с новой версией приложения. В файловом режиме работы отложенные обработчики выполняются сразу, в основном цикле обновления. Отложенные обработчики обновления могут выполняться одновременно в несколько потоков, сокращая длительность обновления.

Настройка

Подготовка к использованию подсистемы

Для библиотеки или конечной конфигурации создать отдельный общий модуль с именем `ОбновлениеИнформационнойБазы<Сокращение>`, где сокращение – короткое имя библиотеки или конфигурации. Пример: общий модуль `ОбновлениеИнформационнойБазыБСП`.

Затем добавить имя созданного модуля в процедуру `ПриДобавленииПодсистем` общего модуля `ПодсистемыКонфигурацииПереопределяемый`.

Из общего модуля `ОбновлениеИнформационнойБазыБСП` скопировать определения следующих процедур (очистив их содержимое) в созданный общий модуль:

```
Процедура ПриДобавленииПодсистемы(Описание) Экспорт

КонецПроцедуры
Процедура ПриДобавленииОбработчиковОбновления(Обработчики) Экспорт

КонецПроцедуры
Процедура ПередОбновлениемИнформационнойБазы() Экспорт

КонецПроцедуры
Процедура ПослеОбновленияИнформационнойБазы(Знач ПредыдущаяВерсия, Знач ТекущаяВерсия,
        Знач ВыполненныеОбработчики, ВыводитьОписаниеОбновлений, МонопольныйРежим) Экспорт

КонецПроцедуры
Процедура ПриПодготовкеМакетаОписанияОбновлений(Знач Макет) Экспорт

КонецПроцедуры
```

Для конечных конфигураций также добавить три процедуры:

- `ПриДобавленииОбработчиковПерехода СДругойПрограммы`,
- `ПриОпределенииРежимаОбновленияДанных`,
- `ПриЗавершенииПерехода СДругойПрограммы`:

```
Процедура ПриДобавленииОбработчиковПерехода СДругойПрограммы(Обработчики) Экспорт

КонецПроцедуры
Процедура ПриОпределенииРежимаОбновленияДанных(РежимОбновленияДанных, СтандартнаяОбработка) Экспорт

КонецПроцедуры
Процедура ПриЗавершенииПерехода СДругойПрограммы(Знач ПредыдущееИмяКонфигурации, Знач ПредыдущаяВерсияКонфигурации, Параметры) Экспорт

КонецПроцедуры
```

Пример реализации см. в общем модуле `_ДемоОбновлениеИнформационнойБазыБСП` демонстрационной конфигурации.

Затем в процедуру `ПриДобавленииПодсистемы` вписать имя и версию библиотеки или конфигурации, а также зависимости от других библиотек (если предусмотрены). По указанным зависимостям вычисляется порядок вызова обработчиков обновления данных библиотек. Если зависимости не указаны, порядок вызова обработчиков библиотек определяется порядком добавления имен модулей библиотек в общем модуле `ПодсистемыКонфигурацииПереопределяемый`. При этом библиотека с именем `СтандартныеПодсистемы` будет вызываться первой, а библиотека, у которой имя совпадает со значением свойства `Метаданные.Имя`, всегда будет вызываться последней.

```
// См. описание этой же процедуры в модуле ОбновлениеИнформационнойБазыБСП.
Процедура ПриДобавленииПодсистемы(Описание) Экспорт

    Описание.Имя = "БиблиотекаСтандартныхПодсистемДемо";
    Описание.Версия = "2.1.3.24";

    // Требуется библиотека стандартных подсистем.
    Описание.ТребуемыеПодсистемы.Добавить("СтандартныеПодсистемы");

КонецПроцедуры
```

При этом в коде процедуры `ПриДобавленииПодсистемы` не следует получать имя и версию напрямую из свойств конфигурации `Метаданные.Имя` и `Метаданные.Версия`. В противном случае при доработке конфигураций потребуется снимать с поддержки и вносить изменения в модуль обновления поставщика.

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема `Настройки программы`, то в командном интерфейсе администратора приложения необходимо разместить следующие объекты метаданных:

- константа `ДетализироватьОбновлениеИБВЖурналеРегистрации`,
- общая форма `ОписаниеИзмененийПрограммы`,
- обработка `РезультатыОбновленияПрограммы`.

См. пример размещения в демонстрационной конфигурации в группе **Обновление версии программы** формы `ПоискИУстановкаОбновлений` обработки `ПанельАдминистрированияБСП`.

Создать общий макет «ОписаниеИзмененийСистемы»

Описание изменений в новой версии готовится разработчиками конфигурации к выпуску каждой версии конфигурации в общем табличном макете `ОписаниеИзмененийСистемы`. Табличный макет может включать в себя столько разделов, сколько версий содержится в истории версий продукта.

Каждый такой раздел состоит из двух областей:

- В область `ШапкаР_П_В_С` помещается текст заголовка **Новое в версии Р.П.В.С**.

- Область `ВерсияР_П_В_С` содержит описание изменений данной версии; может состоять из подразделов или содержать гиперссылки.

Разделы между собой отделяются отступом, который определяется областью `Отступ`.

Для специальной обработки нажатий на гиперссылки, содержащихся в тексте макета, предназначена процедура-обработчик `ПриНажатииНаГиперссылкуВДокументеОписанияОбновлений` общего модуля `ОбновлениеИнформационнойБазыКлиентПереопределяемый`.

Пример заполнения макета можно посмотреть в демонстрационной конфигурации.

> **Важно!**
>
> При первом внедрении макет `ОписаниеИзмененийСистемы` следует создать до выполнения первого запуска конфигурации в режиме **1С:Предприятие**.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным и функциям подсистемы **Обновление версии ИБ** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Запуск приложения для обновления версии ИБ.
2.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Просмотр описания изменения системы.

Использование при разработке конфигурации

Начальное заполнение данных

При первом запуске информационной базы имеется возможность заполнить предопределенные элементы, сформировать классификаторы, предварительно создать элементы различных справочников, содержащих типовые операции.

Необходимо принять решение по поводу состава объектов метаданных конфигурации, для которых требуется начальное заполнение данных. Например, это могут быть справочники с предопределенными элементами, записи классификаторов или списки типовых операций, например, хозяйственные операции. Затем:

1. Включить эти объекты метаданных в состав общего реквизита `ОтредактированныеПредопределенныеРеквизиты`.
2. Перечислить их в процедуре `ПриОпределенииНастроек` модуля `ОбновлениеИнформационнойБазыПереопределяемый`. Например:

```
Процедура ПриОпределенииНастроек(Параметры) Экспорт

    Параметры.ОбъектыСНачальнымЗаполнением.Добавить(Метаданные.Справочники.РолиИсполнителей);

КонецПроцедуры
```

1. Определить в модуле менеджера каждого объекта экспортные процедуры `ПриНастройкеНачальногоЗаполненияЭлементов`, `ПриНачальномЗаполненииЭлементов`, `ПриНачальномЗаполненииЭлемента` по шаблону:

```
// Определяет настройки начального заполнения элементов
//
// Параметры:
//  Настройки - Структура - настройки заполнения
//   * ПриНачальномЗаполненииЭлемента - Булево - Если Истина, то для каждого элемента будет
//       вызвана процедура индивидуального заполнения ПриНачальномЗаполненииЭлемента.
Процедура ПриНастройкеНачальногоЗаполненияЭлементов(Настройки) Экспорт

КонецПроцедуры

// Вызывается при начальном заполнении справочника.
//
// Параметры:
//  КодыЯзыков - Массив - список языков конфигурации. Актуально для мультиязычных конфигураций.
//  Элементы   - ТаблицаЗначений - данные заполнения. Состав колонок соответствует набору реквизитов
//                                 справочника РолиИсполнителей.
//  ТабличныеЧасти - Структура - описание табличных частей объекта, где:
//   * Ключ - Строка - имя табличной части;
//   * Значение - ТаблицаЗначений - табличная часть в виде таблицы значений, структуру которой
//                                  необходимо скопировать перед заполнением.
//
Процедура ПриНачальномЗаполненииЭлементов(КодыЯзыков, Элементы, ТабличныеЧасти) Экспорт

КонецПроцедуры
// Вызывается при начальном заполнении элемента объекта. Если в процедуре ПриНастройкеНачальногоЗаполненияЭлементов свойство ПриНачальномЗаполн
//
// Параметры:
//  Объект                  - Произвольный - заполняемый объект.
//  Данные                  - СтрокаТаблицыЗначений - данные заполнения.
//  ДополнительныеПараметры - Структура - Дополнительные параметры.
//
Процедура ПриНачальномЗаполненииЭлемента(Объект, Данные, ДополнительныеПараметры) Экспорт

КонецПроцедуры
```

> **Примечание** Если в процедуре `ПриНастройкеНачальногоЗаполненияЭлементов` свойство `ПриНачальномЗаполненииЭлемента` установлено в значение `Ложь`, то процедуру `ПриНачальномЗаполненииЭлемента` можно не добавлять.

Затем в процедуре `ПриНачальномЗаполненииЭлементов` вписать код по заполнению таблицы `Элементы`.

Если выполняется заполнение предопределенных элементов, например ролей исполнителей в бизнес процессах и задачах, то для связи предопределенного элемента с данными в коде заполнения необходимо в поле `ИмяПредопределенныхДанных` указать имя предопределенного элемента, как в конфигураторе. Пример кода заполнения справочника `РолиИсполнителей`:

```
// Вызывается при начальном заполнении справочника РолиИсполнителей.
//
// Параметры:
//  КодыЯзыков - Массив - список языков конфигурации. Актуально для мультиязычных конфигураций.
//  Элементы   - ТаблицаЗначений - данные заполнения. Состав колонок соответствует набору реквизитов
//                                 справочника РолиИсполнителей.
//  ТабличныеЧасти - Структура - описание табличных частей объекта, где:
//   * Ключ - Строка - имя табличной части;
//   * Значение - ТаблицаЗначений - табличная часть в виде таблицы значений, структуру которой
//                                  необходимо скопировать перед заполнением.
//
Процедура ПриНачальномЗаполненииЭлементов(КодыЯзыков, Элементы, ТабличныеЧасти) Экспорт

    Элемент = Элементы.Добавить();
    Элемент.ИмяПредопределенныхДанных = "ОтветственныйЗаКонтрольИсполнения";
    Элемент.Наименование = НСтр("ru='Координатор выполнения задач'", ОбщегоНазначения.КодЯзыкаИнформационнойБазы());
    Элемент.ИспользуетсяБезОбъектовАдресации = Истина;
    Элемент.ИспользуетсяСОбъектамиАдресации  = Истина;
    Элемент.ВнешняяРоль                      = Ложь;
    Элемент.Код                              = "000000001";
    Элемент.КраткоеПредставление             = НСтр("ru='000000001'");
    Элемент.ТипыОсновногоОбъектаАдресации = ПланыВидовХарактеристик.ОбъектыАдресацииЗадач.ВсеОбъектыАдресации;

КонецПроцедуры
```

Для создания новых поставляемых элементов из кода, например данных классификаторов, наборов дополнительных свойств и реквизитов, типовых хозяйственных операций и др., следует определить ключевой реквизит, который однозначно идентифицирует элемент. И затем установить имя этого реквизита свойству `ИмяКлючевогоРеквизита` в процедуре `ПриНастройкеНачальногоЗаполненияЭлементов`.

```
// Смотри также ОбновлениеИнформационнойБазыПереопределяемый.ПриНастройкеНачальногоЗаполненияЭлементов
//
// Параметры:
//  Настройки - см. ОбновлениеИнформационнойБазыСлужебный.НастройкиЗаполненияЭлементов
//
Процедура ПриНастройкеНачальногоЗаполненияЭлементов(Настройки) Экспорт

    Настройки.ИмяКлючевогоРеквизита          = "ИмяПредопределенногоВида";

КонецПроцедуры
```

Код заполнения элементов в процедуре `ПриНачальномЗаполненииЭлементов` аналогичен коду заполнения предопределенных элементов. Пример реализации см. в демонстрационной конфигурации в модуле менеджера справочника `ВидыКонтактнойИнформации`.

Существуют ситуации, когда в качестве идентифицирующего элемента необходимо использовать уникальный идентификатор (УИД) ссылки. В таком случае при записи объекта этот УИД будет установлен в качестве ссылки объекта. Для этого в процедуре `ПриНастройкеНачальногоЗаполненияЭлементов` у свойства `ИмяКлючевогоРеквизита` следует установить значение `Ссылка`:

```
// Смотри также ОбновлениеИнформационнойБазыПереопределяемый.ПриНастройкеНачальногоЗаполненияЭлементов
//
// Параметры:
//  Настройки - см. ОбновлениеИнформационнойБазыПереопределяемый.ПриНастройкеНачальногоЗаполненияЭлементов.Настройки
//
Процедура ПриНастройкеНачальногоЗаполненияЭлементов(Настройки) Экспорт

    Настройки.ИмяКлючевогоРеквизита          = "Ссылка";

КонецПроцедуры
А затем в коде процедуры ПриНачальномЗаполненииЭлементов описать заполнение элементов, указав этотУИД :

// См. УправлениеСвойствамиПереопределяемый.ПриНачальномЗаполненииЭлементов
//
// Параметры:
//  КодыЯзыков - см. ОбновлениеИнформационнойБазыПереопределяемый.ПриНачальномЗаполненииЭлементов.КодыЯзыков
//  Элементы   - см. ОбновлениеИнформационнойБазыПереопределяемый.ПриНачальномЗаполненииЭлементов.Элементы
//  ТабличныеЧасти - см. ОбновлениеИнформационнойБазыПереопределяемый.ПриНачальномЗаполненииЭлементов.ТабличныеЧасти
//
Процедура ПриНачальномЗаполненииЭлементов(КодыЯзыков, Элементы, ТабличныеЧасти) Экспорт

// Наборы справочника Демо: Партнеры
    Набор = Элементы.Добавить();
    Набор.ИмяПредопределенногоНабора = "Справочник__ДемоПартнеры";
    Набор.ЭтоГруппа = Истина;
    Набор.Используется = Истина;
    Набор.Ссылка = Новый УникальныйИдентификатор("2c8d6c08-1d35-43ce-a690-32ccf53b03f2");
    МультиязычностьСервер.ЗаполнитьМультиязычныйРеквизит(Набор, "Наименование",
            "ru = 'Демо: Партнеры'", КодыЯзыков); // @НСтр-1

    ДочернийНабор = Элементы.Добавить();
    ДочернийНабор.Родитель = Новый УникальныйИдентификатор("2c8d6c08-1d35-43ce-a690-32ccf53b03f2");
    ДочернийНабор.ИмяПредопределенногоНабора = "Справочник_Партнеры_Клиенты";
    ДочернийНабор.Используется = Истина;
    ДочернийНабор.Ссылка = Новый УникальныйИдентификатор("277e1f06-e4f6-4c23-8d31-0d4347e207a2");
    МультиязычностьСервер.ЗаполнитьМультиязычныйРеквизит(ДочернийНабор, "Наименование",
        "ru = 'Клиент'", КодыЯзыков); // @НСтр-1
КонецПроцедуры
```

Пример реализации см. в демонстрационной конфигурации в общем модуле `_ДемоСвойства` и модуле менеджера справочника `НаборыДополнительныхРеквизитовИСведений`.

Для случаев, когда необходимо расширить состав поставляемых данных у объектов метаданных, стоящих на поддержке других библиотек, или когда для разных справочников используется общий механизм заполнения, то код заполнения следует размещать в одноименных процедурах общего модуля `ОбновлениеИнформационнойБазыПереопределяемый`.

Для автоматического формирования кода процедуры `ПриНачальномЗаполненииЭлементов` рекомендуется использовать инструмент **Начальное заполнение данных**, который входит в состав дистрибутива библиотеки в виде внешней обработки `НачальноеЗаполнениеДанных.epf`.

В первый раз следует:

- выбрать объект метаданных, его реквизиты, которые необходимо заполнить;
- переключиться на вариант **Редактировать в коде** и скопировать автоматически сгенерированный код в процедуру `ПриНачальномЗаполненииЭлементов`.

В дальнейшем для редактирования кода следует:

- скопировать его в инструмент **Начальное заполнение данных** из процедуры `ПриНачальномЗаполненииЭлементов`;
- переключиться на вариант редактировать таблице и добавить новые строки и отредактировать существующие;
- переключиться на вариант редактировать в коде и скопировать автоматически сгенерированный код в процедуру `ПриНачальномЗаполненииЭлементов`.

Для обновления предопределенных данных при переходе с версии на версию рекомендуется создать отложенный обработчик обновления, где:

- В процедуре регистрации элементов добавить вызов процедуры `ОбновлениеИнформационнойБазы.ЗарегистрироватьПредопределенныеЭлементыДляОбновления` указав один из вариантов регистрации элементов: все элементы, новые и измененные, только мультиязычные измененные строки;
- Затем в процедуре обработки данных для перехода на новую версию вызвать `ОбновлениеИнформационнойБазы.ЗаполнитьЭлементыНачальнымиДанными` для заполнения элемента данными, описанными в процедуре `ПриНачальномЗаполненииЭлементов` модуль менеджера объекта;
- В таком случае значения реквизитов будут безусловно восстановлены по данным начального заполнения. Если требуется восстановить данные с учетом изменений, внесенных пользователями, необходимо написать собственный отложенный обработчик обновления.

Пример обработчика перехода см. в общем модуле `_ДемоОбновлениеИнформационнойБазыБСП` демонстрационной конфигурации для справочника `РолиИсполнителей`.

Разработка обработчиков обновления

Для подключения своих обработчиков обновления необходимо в модуль обновления информационной базы библиотеки или конфигурации, созданный на предыдущем шаге, добавить описание обработчиков обновления в процедуру `ПриДобавленииОбработчиковОбновления`.

Для каждого обработчика обновления нужно добавить фрагмент 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"};
```

> Для формирования описания обработчика обновления рекомендуется воспользоваться инструментом **Описание обработчиков обновления** (в меню **Функции для технического специалиста**). После завершения можно получить текстовое описание обработчика и разместить его в указанной процедуре.

Строка таблицы значений `Обработчик` имеет различный состав полей, который зависит от вида обработчика обновления: монопольный, оперативный или отложенный. Свойства, общие для всех видов обработчиков обновления:

- `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 в качестве версии указан символ «*», то обработчик обновления должен выполняться каждый раз при обновлении информационной базы независимо от номера версии конфигурации;
  - если свойство `Версия` не задано, то должно быть установлено в `Истина` свойство `НачальноеЗаполнение` (см. далее).
- `Процедура` (`Строка`) – имя процедуры обновления. Процедура обновления должна располагаться в серверном общем модуле и должна быть экспортной. Например, `ОбновлениеИБ.ПерейтиНаВерсию_1_2_3_4`.
- `НачальноеЗаполнение` (`Булево`) – если `Истина`, то процедура обновления будет вызвана при первом запуске на пустой информационной базе (версия «0.0.0.0»), созданной из файла поставки конфигурации и не содержащей данных. Это обработчики первоначального заполнения базы. По умолчанию – `Ложь`.
- `РежимВыполнения` (`Строка`) – принимает одно из значений: `Монопольно`, `Оперативно` и `Отложенно`. Если свойство не задано, то обработчик – монопольный.

Например, для выполнения двух экспортных процедур `ВыполнятьВсегдаПриСменеВерсии` и `ПерейтиНаВерсию_1_0_0_5` общего модуля `_ДемоОбновлениеИнформационнойБазыБСП` при переходе с версии 1.0.0.1 на версию 1.0.0.5 необходимо разместить следующий фрагмент кода в процедуре `ПриДобавленииОбработчиковОбновления`:

```
Процедура ПриДобавленииОбработчиковОбновления(Обработчики) Экспорт
// Подключаются процедуры – обработчики обновления конфигурации.
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.0.0.0";
Обработчик.Процедура = "_ДемоОбновлениеИнформационнойБазыБСП.ПерейтиНаВерсию_1_0_0_0";
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.0.0.5";
Обработчик.Процедура = "_ДемоОбновлениеИнформационнойБазыБСП.ПерейтиНаВерсию_1_0_0_5";
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "*";
Обработчик.Процедура = "_ДемоОбновлениеИнформационнойБазыБСП.ВыполнятьВсегдаПриСменеВерсии";
КонецПроцедуры
```

Как выбрать способ обработки данных: монопольный, оперативный или отложенный

В зависимости от значения свойства `РежимВыполнения` можно выбрать наиболее эффективный способ обработки данных.

1. `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 приложения прерывается.

Монопольные обработчики предназначены для обновления тех данных, обработка которых должна быть обязательно завершена к моменту входа пользователей в приложение. 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.

2. `Оперативно` – если обработчик обновления необходимо выполнять не монопольно: при активных сеансах работы пользователей, регламентных заданий, внешних соединений и подключений через веб-сервисы.

   Оперативные обработчики следует применять в редких случаях, когда важно сократить время ожидания пользователей на обновление информационной базы – например, для обработки неразделенных данных в модели сервиса, инициализации настроек пользователей и т. п. Как правило, их следует применять только при выпуске исправительных релизов. Подробнее об оперативном обновлении см. раздел Оперативное обновление на исправительные релизы конфигураций.

3. `Отложенно` – если обработчик обновления необходимо выполнять в фоне, после того как завершено выполнение монопольных (оперативных) обработчиков и пользователям уже разрешен вход в приложение.

   Отложенные обработчики предназначены для обработки той части данных ИБ, которые не препятствуют пользователям начинать свою работу с новой версией приложения, 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 об отложенной обработке данных см. раздел Отложенное обновление больших архивов данных.

Общие рекомендации по реализации обработчиков обновления

- Код обработчиков обновления выполняется на сервере, поэтому они не должны содержать никакой логики по интерактивному взаимодействию с пользователем.
- В случае критической ошибки при обновлении в коде обработчика необходимо вызвать 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.
- Код обработчика обновления должен быть рассчитан на неоднократное выполнение, чтобы его повторное выполнение не приводило, например, к дублированию данных в информационной базе.
- На одну и ту же версию конфигурации может быть написано сколь угодно много обработчиков. Например, это могут быть обработчики от разных подсистем. При этом порядок их вызова для одной версии может быть произвольным.
- Порядок вызова обработчиков обновления в пределах одной версии является случайным, т. е. нельзя ставить в зависимость работоспособность одного обработчика обновления от выполнения другого обработчика. Если подобные зависимости появляются, то такие обработчики необходимо объединять в один.

Кроме того, обработчик обновления не должен содержать лишних, избыточных действий с данными – должен выполняться максимально быстро. Для этого в большинстве случаев необходимо отключать бизнес-логику при обработке данных, а также отключать регистрацию изменений на узлах планов обмена, чтобы избежать отправки всего объема обработанных данных во всех узлы. Таким образом:

- в распределенной информационной базе (РИБ) обработка данных должна выполняться независимо в каждом из узлов;
- при обмене между различными приложениями 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`.

Таким образом, в коде обработчика обновления вместо кода вида:

```
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();
```

Для сокращения объема кода рекомендуется использовать процедуру `ЗаписатьДанные` общего модуля `ОбновлениеИнформационнойБазы`:

```
ОбновлениеИнформационнойБазы.ЗаписатьДанные(ДокументОбъект);
```

Для удобной отладки монопольных и оперативных обработчиков обновления можно воспользоваться параметром запуска `РежимОтладки` (обновление будет выполнено без использования фоновых заданий) или флажком **Выполнять обновление версии ИБ без установки монопольного режима и без фонового задания (режим отладки)** в обработке `ОбновлениеВерсииИБ`.

Отложенное обновление больших архивов данных

В тех случаях, когда исчерпаны все остальные средства по оптимизации обработчиков обновления и можно выделить некоторые действия по обработке данных, выполнение которых не требуется обязательно для начала работы пользователей с приложением, рекомендуется перенести эту обработку на более поздний момент времени и выполнять ее отложенно.

Отложенная обработка данных не блокирует вход пользователей в приложение и позволяет избежать ситуаций, когда обновление больших баз занимает существенное время (сутки и более), что нарушает график работы компании (из-за большого времени простоя информационной

системы).

> **Важно!**
>
> Отложенная обработка данных возможна только в клиент-серверном варианте работы. В файловом режиме работы отложенные обработчики обновления выполняются сразу, до начала работы пользователей с новой версией приложения.

Рекомендуется реализовать отложенные обработчики обновления для обработки больших архивов данных за закрытые/прошлые периоды, неактивных позиций номенклатуры, закрытых договоров, различных данных, отключенных в данный момент функциональными опциями и т. п. В большинстве случаев отложенно следует обновлять документы, регистры, бизнес-процессы и задачи, которые имеют тенденцию накапливаться со временем.

Механизм отложенной обработки данных имеет два режима выполнения, которые настраиваются отдельно для каждой библиотеки и основной конфигурации:

- **Последовательно** (по умолчанию) – отложенные обработчики обновления выполняются последовательно в интервале от номера версии информационной базы до номера версии конфигурации включительно (по возрастанию номеров версий, которые указаны в обработчиках). До тех пор, пока один из обработчиков не завершил обработку своей порции данных, следующий не запускается. Данный режим обновления подходит для конфигураций (и библиотек), в которых отложенные обработчики для новых версий обрабатывают те же данные, что обрабатывали обработчики более старых версий. Кроме того, за счет последовательного выполнения к ним предъявляются минимальные требования по «устойчивости» к обрабатываемым данным: при обновлении «через» несколько версий они гарантированно могут рассчитывать на определенное начальное состояние обрабатываемых данных, которое осталось после выполнения обработчиков предыдущей версии.
- **Параллельно** – отложенный обработчик после обработки первой порции данных передает управление следующему обработчику, а после выполнения последнего обработчика цикл повторяется заново, пока все данные не будут обработаны. Таким образом одновременно обновляются объекты информационной базы сразу всех типов, в отличие от последовательного режима, при котором объекты разных типов обрабатываются по очереди и многократно (при обновлении «через» несколько версий).

Разработка отложенных обработчиков обновления для обоих режимов существенно различается. Особенности разработки обработчиков для отложенного режима обновления описаны отдельно.

Для того чтобы указать, что обработчик обновления должен выполняться отложенно, необходимо свойству `РежимВыполнения` присвоить значение `Отложенно`, указать уникальный идентификатор и задать комментарий, который кратко поясняет пользователю, какие данные и как он обрабатывает.

Свойства, специфичные для обоих видов отложенных обработчиков обновления:

- `Комментарий` (`Строка`) – комментарий заполняется обязательно и не должен совпадать с комментариями к другим обработчиками обновления. В нем рекомендуется описывать не только суть выполняемого действия, но и масштаб временно неработоспособного функционала, например:
  - Подготовка индекса для поиска отчетов, предусмотренных в приложении. Поиск отчетов временно недоступен.
  - Реструктуризация дополнительных реквизитов и сведений. Рекомендуется воздержаться от их редактирования до завершения обработки.
  - Первоначальный расчет количества нерассмотренных писем по папкам. До завершения обработки всех писем их количество может выводиться некорректно.
  - Заполняются движения по новому регистру `Движения Номенклатура-Контрагент` по документам **Расчет себестоимости товаров**. После выполнения обработки появится возможность формировать отчеты по товарам.
- `Идентификатор` (`УникальныйИдентификатор`) – идентификатор отложенного обработчика, который необходимо заполнять для разрешения конфликтов при переименованиях или переносе в другой модуль процедуры обновления. В таких случаях по идентификатору будет определен новый путь к обработчику, и он успешно завершит обработку данных.
- `БлокируемыеОбъекты` (`Строка`) – полные имена объектов через запятую, которые следует блокировать в пользовательском интерфейсе от редактирования до завершения процедуры обработки данных. Подробнее см. Блокировка необработанных данных в пользовательском интерфейсе.
- `ПроцедураПроверки` (`Строка`) – имя функции, которая дополнительно для переданного объекта определяет, завершена ли для него процедура обработки данных. Если переданный объект обработан, то следует вернуть значение `Истина`. Вызывается из процедуры `ПроверитьОбъектОбработан` общего модуля `ОбновлениеИнформационнойБазы`. Подробнее см. Блокировка необработанных данных в пользовательском интерфейсе.
- `Многопоточный` (`Булево`) – значение `Истина` следует установить обработчикам обновления, которые способны обрабатывать данные сразу в несколько потоков. По умолчанию, `Ложь`. Подробнее см. Рекомендации по многопоточному выполнению обработчиков обновления.
- `Порядок` (`Перечисление.ПорядокОбработчиковОбновления`) - может принимать значения `Критичный`, `Обычный` и `Некритичный`. Подробнее см. [Порядок выполнения отложенных обработчиков обновления].

Например:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.2.3.4";
Обработчик.Процедура = "Заказы.ЗаполнитьСтатусЗаказовПокупателей";
Обработчик.РежимВыполнения = "Отложенно";
Обработчик.Идентификатор = Новый УникальныйИдентификатор("83d5c5dd-1462-4d72-ab98-f8f5dcc0664d");
Обработчик.Комментарий = НСтр("ru = 'Заполняет значение нового реквизита СтатусЗаказа у документов ""Заказ покупателя"" прошлых периодов.'");
```

Синтаксис процедуры – обработчика отложенного обновления:

```
Процедура ЗаполнитьСтатусыЗаказовПокупателей(Параметры) Экспорт
```

где `Параметры` – `Структура` со свойствами:

- `ОбработкаЗавершена` (`Булево`) – для того, чтобы обработчик был вызван повторно для обработки следующей порции данных, следует записать в него значение `Ложь` ;
- `ВерсияПодсистемыНаНачалоОбновления` (`Строка`) - номер версии подсистемы, к которой относится обработчик обновления, на момент начала обновления приложения. Может быть полезно для ветвления логики выполнения обработчика в зависимости от версии, с которой выполняется обновление приложения.
- `ПрогрессВыполнения` (`Структура`) – необходимо заполнять для отображения прогресса обработки данных:
  - `ВсегоОбъектов` (`Число`) – общее количество объектов, которое необходимо обработать;
  - `ОбработаноОбъектов` (`Число`) – сколько объектов уже обработано.
- `ОбновляемыеДанные` (`Структура`) – если у обработчика обновления установлено свойство `Многопоточный` в значение `Истина` , то в этом свойстве в обработчик передается порция обновляемых данных, которая должна быть обработана в текущем потоке. Получать таблицу значений обновляемых данных следует с помощью вызова функции `ОбновлениеИнформационнойБазы.ДанныеДляОбновленияВМногопоточномОбработчике` , а не напрямую.

Кроме того, в структуру `Параметры` можно добавить произвольное количество свойств произвольных типов, значения которых будут автоматически запоминаться между вызовами процедуры – обработчика отложенного обновления. Таким образом можно передавать контекст отложенной обработки между ее вызовами (например, дату, по которую был обработан архив документов). При этом рекомендуется не сохранять в свойствах структуры большие объемы данных, а использовать только примитивные типы (например, `Дата`).

Отложенную обработку данных необходимо выполнять порциями, чтобы не создавать длительную нагрузку на сервер предприятия и СУБД. По умолчанию размер порции – 1000 (документов, записей и т. п.). Размер порции можно увеличить для небольших объектов и уменьшить для документов, в которых большие табличные части (в среднем).

Также рекомендуется начинать обработку с самых свежих данных.

Например:

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ ПЕРВЫЕ 1000
|    ЗаказПокупателя.Ссылка
|ИЗ
|    Документ.ЗаказПокупателя КАК ЗаказПокупателя
|ГДЕ
|    ЗаказПокупателя.СтатусЗаказа = &ПустаяСсылка
|
|УПОРЯДОЧИТЬ ПО
|    ЗаказПокупателя.Дата УБЫВ";
```

Отложенные обработчики обновления должны самостоятельно заботиться о целостности обновляемых данных: при чтении данных с последующим изменением требуется выполнять эти действия в транзакции и устанавливать исключительную управляемую блокировку.

Например:

```
НачатьТранзакцию();
Попытка
    Блокировка = Новый БлокировкаДанных;
    ЭлементБлокировки = Блокировка.Добавить("Документ._ДемоЗаказПокупателя");
    ЭлементБлокировки.УстановитьЗначение("Ссылка", ЗаказПокупателя.Ссылка);
    Блокировка.Заблокировать();

    ДокументОбъект = ЗаказПокупателя.Ссылка.ПолучитьОбъект();

    // Если объект ранее был удален или обработан другими сеансами, пропускаем его.
    Если ДокументОбъект = Неопределено Тогда
        ОтменитьТранзакцию();
        Возврат;
    КонецЕсли;
    Если ДокументОбъект.СтатусЗаказа = Перечисления._ДемоСтатусыЗаказовПокупателей.ПустаяСсылка() Тогда
        ОтменитьТранзакцию();
        Возврат;
    КонецЕсли;

    // Обрабатываем документ
    // ...

    ОбновлениеИнформационнойБазы.ЗаписатьДанные(ДокументОбъект);

    ЗафиксироватьТранзакцию();
Исключение
    ОтменитьТранзакцию();
    ВызватьИсключение;
КонецПопытки;
```

При возникновении исключения в обработчике отложенного обновления следует всегда пробрасывать его в вызывающий код (см. блок `Исключение – КонецПопытки` в примере выше). В противном случае может возникнуть ситуация, когда обработчик будет выполняться бесконечно долго (если параметру `ОбработкаЗавершена` присвоено значение `Ложь`), а данные ИБ не будут обновлены.

При возникновении исключения в обработчике он помечается как ошибочный, и через определенный интервал времени выполняются еще две попытки его запуска. Тем самым обработчик сможет корректно обновить данные, временно заблокированные другими сеансами. Но если обработчик продолжает и дальше прерываться с исключением (допущена ошибка в самом обработчике), то он пропускается и ставится в очередь выполнения только при очередном обновлении версии приложения. Это позволяет обновить данные при переходе на следующий исправительный релиз, в котором исправлена ошибка в обработчике.

В общем случае при обработке данных конкретной таблицы (документа, регистра и т. п.) некоторая часть ее данных требуется пользователям сразу к моменту начала работы в новой версии приложения, а все остальное может быть обработано отложенно. В таких случаях рекомендуется реализовать два обработчика обновления: монопольный и отложенный.

В редких случаях, когда в новой версии конфигурации появился монопольный (или оперативный) обработчик обновления, данные которого обрабатывались в предыдущих версиях отложенно, следует:

- пересмотреть проектное решение и сделать такой обработчик отложенным,
- либо «старые» отложенные обработчики обновления сделать монопольными (оперативными).

В противном случае возникнет ситуация, когда данные будут обработаны в неправильном порядке: сначала выполнится монопольный (оперативный) обработчик, который рассчитывается на то, что данные были обработаны ранее отложенно.

Блокировка необработанных данных в пользовательском интерфейсе

Если отложенный обработчик обрабатывает не только архивные данные, но и данные текущего периода, то для предотвращения некорректной работы приложения рекомендуется блокировать еще не обработанные данные от изменения пользователями приложения, а также отчеты и обработки, которые могут некорректно работать до завершения отложенного обновления этих данных.

Для этого необходимо:

- в свойстве `БлокируемыеОбъекты` отложенного обработчика обновления указать те объекты метаданных, с которыми он работает (читает или записывает), а также связанные с ними отчеты и обработки. Если в конфигурации используется подсистема **Варианты отчетов**, то отчеты в списке блокируемых объектов указывать не требуется. Для них автоматически[[1]](#прим1_1) будут определены читаемые объекты, и если выводимые в отчете данные содержат необновленную информацию, то будет показано соответствующее предупреждение;

  > [1] Для Определение таблиц, используемых отчетом, в некоторых случаях может потребоваться явное указание этих таблиц в самом отчете. Подробнее см. в разделе Варианты отчетов.

- в свойстве `ПроцедураПроверки` указать имя экспортной функции, которая дополнительно проверяет, нужно ли блокировать конкретный объект. Например:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.2.3.4";
Обработчик.Идентификатор = Новый УникальныйИдентификатор("b3be66c5-708d-42c8-a019-818036d09d06");
Обработчик.Процедура = "Заказы.ЗаполнитьСтатусыЗаказовПокупателей";
Обработчик.Комментарий = НСтр("ru = 'Заполнение значения нового реквизита ""Статус заказа"" у документов ""Демо: Заказ покупателя"" прошлых пер
    |До завершения обработки ""Статус заказа"" данных документов будет отображаться некорректно.'");
Обработчик.РежимВыполнения = "Отложенно";
Обработчик.ПроцедураПроверки = "Заказы.ЗаказПокупателяОбработан";
Обработчик.БлокируемыеОбъекты = "Документ.ЗаказПокупателя,Отчет.СтатусыЗаказовПокупателей";
```

Затем для всех объектов, указанных в свойстве `БлокируемыеОбъекты`, необходимо добавить вызов процедуры `ОбновлениеИнформационнойБазы.ПроверитьОбъектОбработан` в обработчике события модуля формы объекта `ПриСозданииНаСервере` и в обработчике события модуля объекта `ПередЗаписью`:

```
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    ОбновлениеИнформационнойБазы.ПроверитьОбъектОбработан(Объект, ЭтотОбъект);
    …
КонецПроцедуры
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

    Если ОбменДанными.Загрузка Тогда
        Возврат;
    КонецЕсли;

    ОбновлениеИнформационнойБазы.ПроверитьОбъектОбработан(ЭтотОбъект);
    …
КонецПроцедуры
```

Функция проверки, указанная в свойстве `ПроцедураПроверки`, возвращает `Истина` для объектов, если объект уже обработан и с ним могут работать пользователи, или `Ложь`, если объект следует заблокировать от редактирования. Она принимает на вход параметр типа `Структура`, свойства которого идентифицируют запрашиваемый объект:

- `Данные` (`ЛюбаяСсылка`, `НаборЗаписей`, `Объект`, `ДанныеФормыСтруктура`) – объект, для которого нужно проверить, обработан ли он данным отложенным обработчиком обновления или еще нет.
- `МетаданныеОбъекта` (`ОбъектМетаданных`) – объект метаданных, соответствующий параметру `Данные`.
- `ПолноеИмя` (`Строка`) – полное имя объекта метаданных.

- `Отбор` (`ЛюбаяСсылка`, `Структура`). Если `Данные` – это ссылочный объект, то значение ссылки; если регистр, подчиненный регистратору – значение отбора по регистратору. Если `Данные` – это независимый регистр сведений, то в этом параметре предается структура, соответствующая установленным отборам по измерениям.
- `ЭтоНовый` (`Булево`). Если `Данные` – это ссылочный объект, то признак нового объекта. Для других типов – всегда `Ложь`.

Ее реализация должна быть достаточно простой, чтобы не сильно замедлять открытие формы, например:

```
Функция ЗаказПокупателяОбработан(Параметры) Экспорт

    Если ТипЗнч(Параметры.Данные) = Тип("ДокументСсылка._ДемоЗаказПокупателя") Тогда
        СтатусЗаказа = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Параметры.Данные, "СтатусЗаказа");
    Иначе
        СтатусЗаказа = Параметры.Данные.СтатусЗаказа;
    КонецЕсли;

    Возврат СтатусЗаказа <> Перечисления._ДемоСтатусыЗаказовПокупателей.ПустаяСсылка();

КонецФункции
```

Пример отложенного обработчика обновления с блокировкой необработанных данных в пользовательском интерфейсе см. в демонстрационной конфигурации: процедура `ПриДобавлении ОбработчиковОбновления` общего модуля `_ДемоОбновлениеИнформационнойБазыБСП`.

Особенности параллельного режима отложенного обновления

Для установки параллельного режима выполнения отложенных обработчиков конфигурации (библиотеки) следует в процедуре `ПриДобавлении Подсистемы` общего модуля `ОбновлениеИнформационнойБазы<Сокращение>` указать:

```
Описание.РежимВыполненияОтложенныхОбработчиков = "Параллельно";
```

Данный режим обновления можно включить с определенной версии, чтобы не пересматривать реализацию старых отложенных обработчиков. Для этого необходимо заполнить свойство `ПараллельноеОтложенноеОбновлениеСВерсии`:

```
Описание.ПараллельноеОтложенноеОбновлениеСВерсии = "2.3.3.20";
```

Свойства, специфичные для отложенных обработчиков обновления с режимом выполнения `Параллельно`:

- `ПроцедураЗаполненияДанныхОбновления` (`Строка`) – указывается процедура, которая регистрирует данные, подлежащие обновлению данным обработчиком. Регистрация выбранных данных выполняется при помощи процедур `ОтметитьКОбработке` и `ОтметитьРегистраторыКОбработке` общего модуля `ОбновлениеИнформационнойБазы`. Например:

  ```
  Процедура ЗарегистрироватьДанныеКОбработкеДляПереходаНаНовуюВерсию(Параметры) Экспорт
      Запрос = Новый Запрос;
      Запрос.Текст =
          "
          ...
          ";
      Результат = Запрос.Выполнить().Выгрузить();
      МассивССылок = Результат.ВыгрузитьКолонку("Ссылка");
      ОбновлениеИнформационнойБазы.ОтметитьКОбработке(Параметры, МассивССылок);
  КонецПроцедуры
  ```

  Эти объекты метаданных следует также включить в состав плана обмена `ОбновлениеИнформационнойБазы`.
- `ПроцедураПроверки` (`Строка`) – всегда указывается `ОбновлениеИнформационнойБазы.ДанныеОбновленыНаНовуюВерсиюПрограммы`.
- `ЗапускатьТолькоВГлавномУзле` (`Булево`) – указать `Истина`, если обработчик обновления может выполняться только в главном узле РИБ. По умолчанию `Ложь`. Может понадобиться, если, например, нужно сгенерировать новые ссылочные объекты в информационной базе. Данные, созданные таким обработчиком обновления, будут отправлены во все дочерние узлы РИБ.
- `ЗапускатьИВПодчиненномУзлеРИБСФильтрами` (`Булево`) – указать `Истина`, если обработчик обновления может выполняться только в подчиненном узле РИБ с фильтрами. По умолчанию `Ложь`. В подчиненном узле РИБ с фильтрами может не быть всех требуемых данных для корректного выполнения обработчика обновления, поэтому данные обрабатываются централизованно в главном узле, после чего отправляются в подчиненные. Включать возможность запуска обработчика нужно после предварительного анализа читаемых и обрабатываемых данных.
- `ЧитаемыеОбъекты` (`Строка`) – список полных имен объектов через запятую, которые обработчик обновления читает при обработке данных.
- `ИзменяемыеОбъекты` (`Строка`) – список полных имен объектов через запятую, которые обработчик обновления изменяет при обработке данных. С помощью значений свойств `ЧитаемыеОбъекты` и `ИзменяемыеОбъекты` контролируются возможные конфликты между обработчиками, изменяющими или читающими одни и те же объекты.
- `ОчередьОтложеннойОбработки` (`Число`) – рассчитывается автоматически.
- `Приоритеты` (`ТаблицаЗначений`) – рассчитывается автоматически. Таблица зависимостей между обработчиками, изменяющими или читающими одни и те же объекты:
  - `Порядок` (`Строка`) – порядок выполнения обработчика относительно другого. Может принимать значения `До`, `После` или `Любой`;
  - `Идентификатор` (`УникальныйИдентификатор`) – идентификатор процедуры, с которой указывается взаимосвязь;
  - `Процедура` (`Строка`) – полное имя процедуры, с которой указывается взаимосвязь.
    Допускается указывать только одно из свойств – `Процедура` или `Идентификатор`.

Например:

```
Обработчик.ЧитаемыеОбъекты = "Документ.ЗаказПокупателя";
Обработчик.ИзменяемыеОбъекты = "Документ.ЗаказПокупателя";
Обработчик.ПриоритетыВыполнения = ОбновлениеИнформационнойБазы.ПриоритетыВыполненияОбработчика();
Приоритет = Обработчик.ПриоритетыВыполнения.Добавить();
Приоритет.Порядок = "До";
Приоритет.Идентификатор = Новый УникальныйИдентификатор("b3be66c5-708d-42c8-a019-818036d09d06");
Приоритет.Процедура = "Документы.ЗаказПокупателя.ОбработатьДанныеДляПереходаНаНовуюВерсию"
```

Приоритеты выполнения обработчиков следует заполнять только в тех случаях, когда по объективным причинам автоматически рассчитанные приоритеты не удовлетворяют решаемой задаче:

1. Обновление данных выполняется в следующем порядке по типам метаданных:
   - Константа
   - Справочник
   - ПланВидовХарактеристик
   - ПланСчетов
   - ПланВидовРасчетов
   - ПланОбмена
   - БизнесПроцесс
   - Задача
   - РегистрСведений (независимый)
   - Документ
   - ЖурналДокументов
   - РегистрСведений (подчиненный регистратору)
   - РегистрНакопления
   - РегистрРасчета
   - РегистрБухгалтерии

   > Эти приоритеты можно переопределить в процедуре `ОбновлениеИнформационнойБазыПереопределяемый.ПриЗаполненииПриоритетовТиповМетаданных` или для конкретного объекта метаданных задать свой порядок, отличный от порядка его типа.

2. Каждому обработчику обновления присваивается порядок обновления, который равен минимальному порядку обновления из всех изменяемых им типов. Например если обработчик изменяет одновременно справочник и документ, то его порядок обновления будет равен порядку обновления справочника.
3. Приоритеты выполнения устанавливаются по принципу:
   - При конфликте чтения или записи двух обработчиков обновляющих разные типы метаданных первым выполняется тот, который изменяет данные с меньшим порядком обновления (сначала константы, справочники, план видов характеристик и т.д.)
   - При конфликте двух обработчиков обновляющих данные одного типа:
     - первым выполняется тот, который изменяет читаемые данные
     - в случае если оба обработчика изменяют друг другу читаемые данные, то порядок любой (предполагается, что обработчики изменяют независимые друг для друга наборы данных, в противном случае это зацикливание порядка выполнения, когда первый обработчик ждет второй, а второй ждет первый)
     - в случае повторной записи порядок выполнения любой (предполагается, что пишутся или разные наборы данных или разные реквизиты в одном и том же объекте, иначе обработчики должны быть объединены в один)

В итоге синтаксис добавления обработчика для параллельного режима отложенного обновления будет иметь следующий вид:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.2.3.4";
Обработчик.Идентификатор = Новый УникальныйИдентификатор("b3be66c5-708d-42c8-a019-818036d09d06");
Обработчик.Процедура = "Документы.ЗаказПокупателя.ОбработатьДанныеДляПереходаНаНовуюВерсию";
Обработчик.Комментарий = НСтр("ru = 'Заполнение значения нового реквизита ""Статус заказа"" у документов ""Заказ покупателя"" прошлых периодов.
    |До завершения обработки ""Статус заказа"" данных документов будет отображаться некорректно.'");
Обработчик.РежимВыполнения = "Отложенно";
Обработчик.ПроцедураЗаполненияДанныхОбновления = "Документы.ЗаказПокупателя.ЗарегистрироватьДанныеКОбработкеДляПереходаНаНовуюВерсию";
Обработчик.ЧитаемыеОбъекты      = "Документ.ЗаказПокупателя";
Обработчик.ИзменяемыеОбъекты    = "Документ.ЗаказПокупателя";
Обработчик.ПроцедураПроверки    = "ОбновлениеИнформационнойБазы.ДанныеОбновленыНаНовуюВерсиюПрограммы";
Обработчик.БлокируемыеОбъекты   = "Документ.ЗаказПокупателя";
```

Процедура, указанная в поле `ПроцедураЗаполненияДанныхОбновления` принимает один параметр типа `Структура` со следующими полями:

- `ПараметрыВыборки` (`Структура`) – параметры, которые механизм обновления использует для выборки данных, которые затем передает в потоки многопоточных обработчиков обновления (присутствует только у многопоточных обработчиков).
- `ЗаписьИзмененийДляПодчиненногоУзлаРИБСФильтрами` (`ЗаписьFastInfoset`) – служебный параметр. Не модифицировать.
- `ИмяФайлаСИзменениями` (`Строка`) – служебный параметр. Не модифицировать.
- `Очередь` (`Число`) – служебный параметр. Не модифицировать.
- `ПовторнаяРегистрация` (`Булево`) – служебный параметр. Не модифицировать.

Многопоточным обработчикам обновления необходимо настраивать параметры выборки данных в процедуре, указанной в поле `ПроцедураЗаполненияДанныхОбновления`, устанавливая необходимые значения в поле `ПараметрыВыборки` (`Структура`), имеющее следующую структуру:

- `ПолныеИменаОбъектов` (`Строка`) – полные имена обновляемых объектов (например, документов), разделенных запятыми.
- `ПолныеИменаРегистров` (`Строка`) – полные имена регистров, разделенных запятыми.
- `ПоляУпорядочиванияПриРаботеПользователей` (`Массив`) – поля упорядочивания, используемые при обновлении с приоритетом работы пользователей.
- `ПоляУпорядочиванияПриОбработкеДанных` (`Массив`) – поля упорядочивания, используемые при обновлении с приоритетом обработки данных.
- `СпособВыборки` (`Строка`) – один из способов выборки:
  - `ОбновлениеИнформационнойБазы.СпособВыборкиИзмеренияНезависимогоРегистраСведений()`;
  - `ОбновлениеИнформационнойБазы.СпособВыборкиРегистраторыРегистра()`;
  - `ОбновлениеИнформационнойБазы.СпособВыборкиСсылки()`.
- `ОптимизироватьВыборкуПоСтраницам` (`Булево`) – если установлено значение `Истина` (по умолчанию), то выборка выполняется несколькими запросами (как в динамическом списке, без компоновки условий по `ИЛИ`). Если установлено значение `Ложь`, то выборка выполняется одним запросом, компонуя условия по `ИЛИ`. Значение `Ложь` рекомендуется использовать, только если запрос выборки данных для обновления работает медленно и его нельзя ускорить (оптимизировать).

Порядок выполнения отложенных параллельных обработчиков обновления

При отложенном параллельном обновлении все данные в приложении обрабатываются равномерно. Однако часть этих данных может быть более критична для начала работы, например, справочники и регистры, без которых невозможен ввод новых документов, чем, например, данные архивных периодов. Для того чтобы ускорить обработку важных данных и быстрее разблокировать основные функции приложения, рекомендуется упорядочить обработку данных по критичности.

Для этого нужно определить, какие обработчики обновляют важные данные, и в описании обработчика обновления указать свойство `Порядок`: `Критичный`, `Обычный` или `Некритичный`. Если порядок не указан, то по умолчанию порядок `Обычный`. Пример:

```
Обработчик = Обработчики.Добавить();
...
Обработчик.Порядок = Перечисления.ПорядокОбработчиковОбновления.Критичный;
```

При этом обработчики обновления будут сгруппированы по порядку, и обработка менее критичных данных начнется не ранее того, как все более критичные данные будут обработаны. Для более равномерного обновления областей в режиме сервиса это произойдет, когда обработка более критичных данных завершится во всех областях.

Обработку отложенных параллельных обработчиков с порядком `Обычный` (например, документов) рекомендуется разделить на обработку свежих данных (они будут обрабатываться вместе с другими обычными обработчиками) и на обработку архивных данных, которые должны быть обработаны позднее. Для этого в процедуре регистрации данных к обновлению необходимо заполнить критерий отбора свежих данных, заполнив в структуре параметров `Параметры` свойство `АктуальныеДанные`:

```
Процедура ЗарегистрироватьДанныеКОбработкеДляПереходаНаНовуюВерсию(Параметры) Экспорт
    ...
    АктуальныеДанные = Параметры.АктуальныеДанные;
    АктуальныеДанные.ПолеОтбора = "Дата";
    АктуальныеДанные.ВидСравнения = ВидСравнения.Больше;
    АктуальныеДанные.Значение = Дата("20220712");
    ...
КонецПроцедуры
```

Описание свойства `АктуальныеДанные` см. в процедуре `ПараметрыВыборкиАктуальныхДанных` общего модуля `ОбновлениеИнформационнойБазы`. Данные, которые удовлетворяют заданному условию, будут обработаны с порядком `Обычные`, а оставшиеся - с порядком `Некритичные`.

Рекомендации по многопоточному выполнению обработчиков обновления

Для выполнения длительных отложенных параллельных обработчиков обновления рекомендуется включать многопоточный режим. Для этого в процедуре `ПриОпределенииНастроек` общего модуля `ОбновлениеИнформационнойБазыПереопределяемый` следует установить параметры:

- `МногопоточноеОбновление` – разрешает многопоточное обновление, если установлено значение `Истина`;
- `КоличествоПотоковОбновленияИнформационнойБазыПоУмолчанию` – количество одновременно выполняющихся обработчиков обновления по умолчанию. Рекомендуемое значение равно наиболее вероятному количеству ядер процессора сервера **1С:Предприятия** у пользователей. Например:

```
Процедура ПриОпределенииНастроек(Параметры) Экспорт

    Параметры.Вставить("МногопоточноеОбновление", Истина);
    Параметры.Вставить("КоличествоПотоковОбновленияИнформационнойБазыПоУмолчанию", 8);
КонецПроцедуры
```

Если включена возможность многопоточного обновления, то помимо конкурентной работы в сеансах пользователей и регламентных заданий одновременно выполняются сразу несколько обработчиков обновления. Поэтому перед каждым выпуском прикладной конфигурации рекомендуется тестировать обновление «большой» реалистично наполненной информационной базы с минимальной поддерживаемой версии в режиме обработки данных с количеством потоков, установленным по умолчанию (8).

Для анализа и устранения выявленных проблем производительности (ожидания на блокировках, таймауты и взаимоблокировки) рекомендуется воспользоваться конфигурацией **Центр управления производительностью**.

Параллельное отложенное обновление в РИБ с фильтрами

Разработка отложенного обновления имеет ряд особенностей в конфигурациях, предусматривающих работу в распределенной информационной базе (РИБ), когда данные синхронизируются не полностью (например, по организациям и т.п.). Такая организация работы называется РИБ с фильтрами.

В таком режиме обработчики отложенного обновления выполняются только в главном узле, т. к. в подчиненном узле может не быть всех требуемых для корректного обновления данных. Исключение составляют обработчики, которым установлен признак `ЗапускатьИВПодчиненномУзлеРИБСФильтрами`. При разработке отложенных обработчиков данный признак рекомендуется использовать в тех случаях, когда точно известно, что обновляемые данные самодостаточны, т. е. могут быть корректно обновлены в любом узле РИБ.

До того момента, как обновление конфигурации будет загружено в подчиненный узел, пользователи могут продолжать вводить данные на старой версии конфигурации подчиненного узла, которые будут отправлены в главный узел и могут привести к появлению некорректных (не обновленных) данных. Чтобы исключить эту ситуацию, после первой синхронизации с каждым подчиненным узлом все параллельные отложенные обработчики в главном узле перезапускаются для обработки полученных из подчиненного узла данных.

При этом стоит учитывать, что несмотря на то, что сами отложенные обработчики в подчиненном узле РИБ не выполняются, процедуры регистрации обновляемых данных для них будут выполнены. Это необходимо для того, чтобы корректно работал механизм защиты от изменения пользователем не обновленных данных.

Отладка обработчиков обновления

Для отладки монопольных и оперативных обработчиков обновления необходимо:

- если у обработчика задана конкретная версия, то:
  - в режиме **Предприятия** открыть регистр сведений `ВерсииПодсистем` (через **Функции для технического специалиста** или перейти по ссылке `e1cib/list/РегистрСведений.ВерсииПодсистем`);
  - выполнить команду **Еще - Включить возможность редактирования** и указать предыдущий номер версии для записи конфигурации;
  - если обработчик относится не к конфигурации, а к библиотеке, то также понизить номер версии и для записи этой библиотеки;
- в конфигураторе поставить точку останова в интересующем обработчике;
- запустить конфигурацию с параметром `/C РежимОтладки`.

Отладка отложенных обработчиков обновления в файловой информационной базе не отличается. Для клиент-серверной информационной базы необходимо:

- выполнить все действия, указанные выше;
- перейти в **Результаты обновления и дополнительная обработка данных** (раздел **Администрирование - Обслуживание - Обновление программы** или перейти по ссылке `e1cib/app/Обработка.РезультатыОбновленияПрограммы`) и затем по гиперссылке внизу открыть окно **Дополнительные процедуры обработки данных**;
- в строке с интересующим обработчиком нажать **Запустить процедуру** или внизу формы нажать кнопку **Запустить**.

Если отложенные обработчики обновления успевают выполниться до того, как нажимаем кнопку **Запустить процедуру**, следует предварительно заблокировать в информационной базе выполнение регламентных заданий.

В процессе разработки для всех обработчиков обновления допустимо в качестве версии указывать значение `ОтладкаОбработчика`:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "ОтладкаОбработчика";
Обработчик.РежимВыполнения = "Оперативно";
Обработчик.Процедура = "ПользователиСлужебный.ПеренестиНастройкиДлиныИСложностиПаролейКонфигуратора";
```

В таком случае обработчик будет выполняться при каждом обновлении и для его отладки достаточно запустить приложение с ключом `ЗапуститьОбновлениеИнформационнойБазы`. Так же это может быть удобно для случаев, когда точный номер версии для обработчика не известен до его помещения в основное хранилище.

> **Примечание**
>
> Кроме понижения номера версии в регистре сведений `ВерсииПодсистем` также можно увеличивать номер версии в свойстве `Версия` конфигурации.
>
> Подробнее про параметр запуска `ЗапуститьОбновлениеИнформационнойБазы` см. в разделе «Приложение 2. Доступные параметры запуска приложения» документации.

Контроль зацикливания обработки данных

Отложенные обработчики обновления вызываются многократно и выбирают данные порциями. Если из-за ошибки в запросе повторно выбираются уже обновленные данные или обработчик зацикливается по другой причине, то обновление информационной базы не зависает, а останавливается с сообщением:

> Превышено допустимое количество запусков процедуры обновления.

Нештатные ситуации с зацикливанием обработчиков обновления контролируются автоматически двумя способами:

- Контроль максимального количества запусков процедуры обработчика.

  Максимальное количество запусков одного обработчика обновления составляет 10000, однако может быть автоматически увеличено для многопоточных обработчиков при большом количестве потоков. Данное ограничение защищает от случаев, когда обработчик повторно

пытается обрабатывать одни и те же данные. Например, это случается из-за ошибки в запросе последовательного отложенного обработчика обновления.

- Контроль холостого запуска обработчика.

  После выполнения каждого обработчика проверяется, были ли обработаны данные. Если никакие данные не были обработаны, то увеличивается количество попыток запуска. После трех холостых запусков обработчик считается зациклившимся и ему устанавливается статус `Ошибка`. Лимит в три попытки может быть автоматически увеличен, если обработчик многопоточный.

Для монопольных и оперативных обработчиков обновления контроля зацикливания не предусмотрено, т.к. они выполняются не порционно за один вызов процедуры обработчика. При зацикливании таких обработчиков обновление информационной базы зависает.

Оперативное обновление на исправительные релизы конфигураций

При разработке исправительных релизов конфигураций рекомендуется сохранять возможность оперативного (динамического) обновления таким образом, чтобы при обновлении на такие версии не требовалось обязательного прекращения работы пользователей и других активных сеансов. Для этого исправительный релиз конфигурации должен удовлетворять следующим условиям:

- в него могут быть внесены только «незначительные» изменения, которые позволяют динамически обновлять конфигурацию базы данных (т. е. когда не требуется реструктуризация информационной базы);
- обработчики обновления версии ИБ не должны требовать установки монопольного режима для своего выполнения;
- изменения в конфигурации должны ограничиваться только теми изменениями, которые не потребуют выполнения обязательных обработчиков обновления, предусмотренных в БСП (свойство `Версия` = «*»).

Для обязательных обработчиков обновления (свойство `Версия` = «*») дополнительно предусмотрена возможность программно определять, действительно ли для их выполнения требуется монопольный режим. Для этого следует установить свойство обработчика `МонопольныйРежим` в значение `Истина`. В этом случае:

- 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`. При этом код обработчика не должен модифицировать данные ИБ.
- Если в ходе выполнения обработчика возникает необходимость внесения изменений в ИБ, обработчик должен установить значение свойства в значение `Истина` и прекратить свое выполнение.
- При втором вызове в режиме выполнения свойство `МонопольныйРежим` содержит значение `Истина`. В этом случае код обработчика может модифицировать данные ИБ. Изменение значения свойства в этом случае игнорируется.

Оперативные обработчики обновления должны самостоятельно заботиться о целостности обновляемых данных: если выполняется более одного изменения данных, требуется выполнять действия в транзакции, а если производится чтение данных с последующим изменением, то требуется устанавливать исключительную управляемую блокировку.

Пример обязательного обработчика обновления с проверкой необходимости монопольного режима см. в демонстрационной конфигурации: процедура `ВыполнятьВсегдаПриСменеВерсии` общего модуля `_ДемоОбновлениеИнформационнойБазыБСП`.

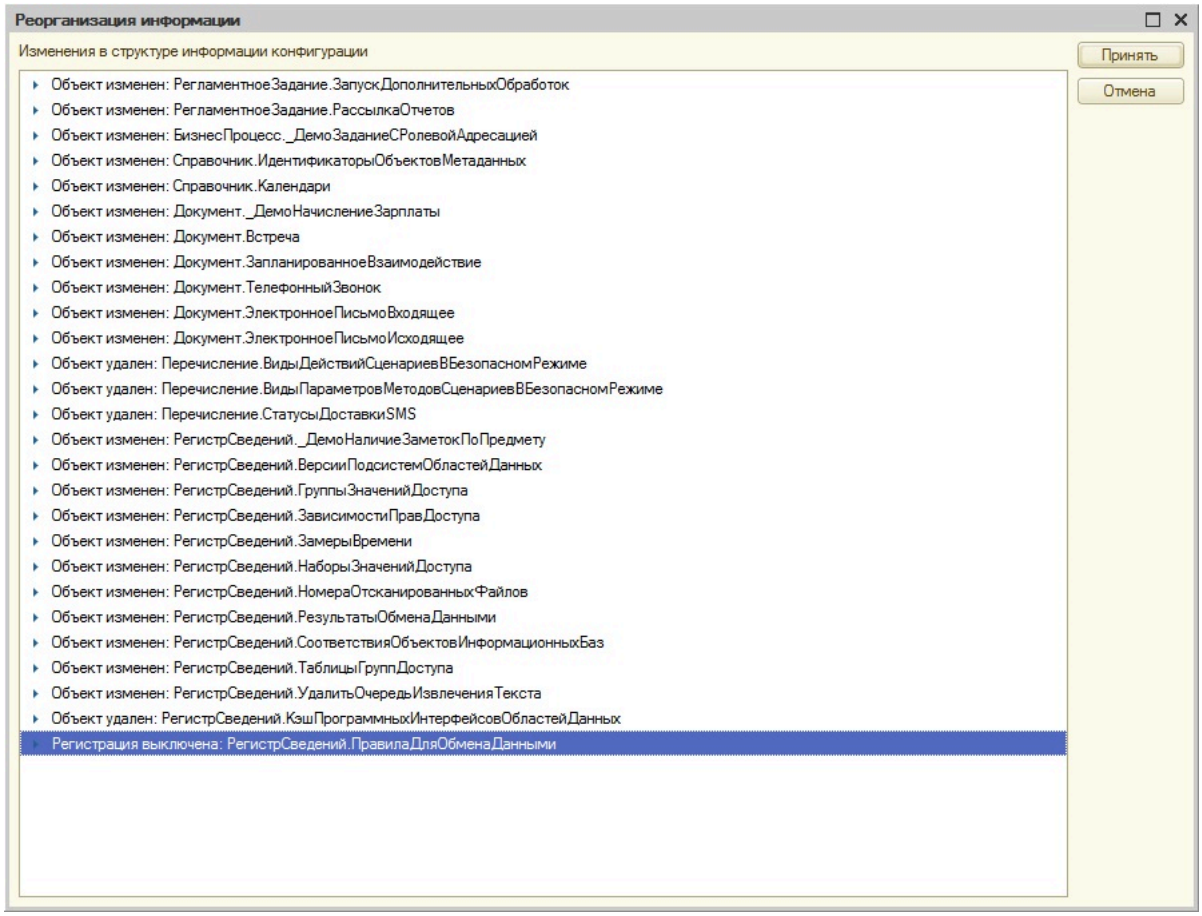
Для гарантированной проверки возможности оперативного обновления на новую версию конфигурации следует выполнить динамическое обновление конфигурации базы данных при активном сеансе **1С:Предприятия** и выполнить запуск нового сеанса администратора. Для этого необходимо:

1. Установить конфигурацию предыдущего релиза.
2. Запустить ИБ предыдущего релиза конфигурации под каким-либо пользователем.
3. Открыть ИБ предыдущего релиза конфигуратором и нажать **F7**.
4. Проверить, что конфигуратор предлагает обновиться динамически. Если не предлагает, см. ниже **Проблема 1**.
5. Обновить динамически, затем выполнить запуск под администратором по **F5**.
6. Проверить, что запуск прошел без ошибок и выводится окно с описанием изменений в новой версии. Если возникли ошибки, см. ниже **Проблема 2**.

Проблема 1. Не проходит динамическое обновление конфигурации.

Для расследования нужно:

- завершить сеанс пользователя (запущенный на шаге 2) и нажать **F7** еще раз;
- будет выведен список объектов конфигурации, препятствующих динамическому обновлению:



Для дальнейшего расследования причин (разработчиками) нужно сравнить тестируемую конфигурацию с предыдущей версией и выяснить, какие именно изменения блокируют динамическое обновление.

Проблема 2. Администратор не может войти в приложение.

Помимо «просто» ошибок в обработчиках обновления может выдаваться ошибка вида:

- Невозможно выполнить обновление информационной базы:
  - невозможно установить монопольный режим,
  - версия конфигурации не предусматривает обновление без установки монопольного режима.

Она означает:

- что один из обработчиков обновления требует монопольного режима для своего выполнения;
- либо что в конфигурацию были внесены изменения, которые потребовали выполнения обязательных обработчиков обновления, предусмотренных в БСП (для версии «*»).

Дальнейшее расследование следует выполнять под отладкой – либо, если была включена константа **Детализировать обновление ИБ в Журнале регистрации**, то информацию о проблемном обработчике можно посмотреть в журнале регистрации.

Переход с другой конфигурации

Кроме переходов между версиями одной конфигурации подсистема позволяет также выполнять прикладной код при переходе с других конфигураций в режиме обновления. Например, можно предусмотреть переход с базовой версии конфигурации на версию ПРОФ, с ПРОФ на КОРП или другие переходы, при которых меняется не только номер версии, но и имя конфигурации. Если при этом добавляются новые подсистемы, то для них автоматически будут выполнены оперативные и монопольные обработчики начального заполнения.

При необходимости выполнить прикладной код при таком переходе в процедуре `ПриДобавленииОбработчиковПерехода СДругойПрограммы` общего модуля, имя которого задано в процедуре `ПодсистемыКонфигурацииПереопределяемый.ПриДобавленииПодсистем`, следует определить обработчики перехода в виде:

```
Обработчик = Обработчики.Добавить();
Обработчик.ПредыдущееИмяКонфигурации = "УправлениеТорговлей";
Обработчик.ОбщиеДанные = Ложь;
Обработчик.Процедура = "ОбновлениеУПП.ЗаполнитьУчетнуюПолитику";
```

Для ветвления логики перехода в зависимости от версии исходной конфигурации внутри обработчика можно воспользоваться функцией `ВерсияИБ` общего модуля `ОбновлениеИнформационнойБазы`.

Для поддержки перехода с другой конфигурации в режиме сервиса обработчикам перехода необходимо указывать свойство `ОбщиеДанные = Истина` для обработчиков неразделенных данных. Если обработчик обновляет разделенные данные, то указывать данное свойство необязательно.

Пример обработчика перехода см. в общем модуле `_ДемоОбновлениеИнформационнойБазыБСП` демонстрационной конфигурации.

Особенности обработчиков обновления при работе в модели сервиса

Схема выполнения обработчиков обновления версии ИБ в модели сервиса состоит из двух частей:

- **Выполняется обновление общих (неразделенных) данных**. При необходимости выполняется блокировка всей информационной базы (используется монопольный режим). Порядок выполнения обработчиков обновления:
  - выполняются обработчики с признаком `ОбщиеДанные = Истина`;
  - устанавливаются блокировки на все области данных;
  - в очередь заданий добавляются задания на выполнение во всех областях данных обработчиков обновления с признаком `ОбщиеДанные = Ложь`.
- **Выполняется обновление разделенных данных (во всех областях данных)**. Выполняется механизмом очереди заданий. При необходимости используется блокировка области данных. Особенности выполнения:
  - при обновлении существующих областей данных пропускаются обязательные обработчики обновления (`Версия` = «*»). Чтобы эти обработчики были выполнены, необходимо их повторно зарегистрировать в обработчике обновления общих данных, у которого свойство `УправлениеОбработчиками` равно `Истина`;
  - при подготовке новых областей данных обязательные обработчики обновления разделенных данных всегда выполняются.

Свойства, специфичные только для обработчиков обновления в модели сервиса:

- `ОбщиеДанные` (`Булево`) – если `Истина`, то обработчик обновления будет выполняться из неразделенного сеанса до выполнения любых обработчиков обновления разделенных данных с признаком `ОбщиеДанные = Ложь`. Такой обработчик может обращаться только к неразделенным данным. Допустимо указывать только для обработчиков с режимом выполнения `Монопольно` и `Оперативно`. Если указать значение `Истина` для обработчика с режимом выполнения `Отложенно`, будет выдано исключение. По умолчанию `Ложь`.
- `УправлениеОбработчиками` (`Булево`) – если `Истина`, в процедуру обработчика обновления будет передан дополнительный параметр типа `Структура` со свойством `РазделенныеОбработчики`. Свойство содержит дополнительную таблицу обработчиков, в которую нужно добавить обработчики разделенных данных, как и в основную таблицу `Обработчики`. По умолчанию `Ложь`.

Пример обработчика обновления общего (неразделенного) классификатора банков:

```
Обработчик = Обработчики.Добавить();
Обработчик.Версия = "1.2.3.4";
Обработчик.Процедура = "КлассификаторБанков.ЗаполнитьИНН";
Обработчик.ОбщиеДанные = Истина;
```

Подключение обработчиков обновления в доработанных конфигурациях

При доработке типовых конфигураций часто возникает необходимость добавлять свои документы и справочники. В дальнейшем это приводит также к необходимости создавать свои обработчики обновления. Для этих целей следует использовать следующий подход.

Считаем, что исходная типовая конфигурация становится библиотекой со своими собственными номером версии и обработчиками обновления, а доработанная конфигурация выступает в качестве основной конфигурации с собственным именем и своей нумерацией версий, разработанной на базе этой библиотеки.

Например, в типовой конфигурации `БухгалтерияПредприятияКОРП` в общем модуле `ПодсистемыКонфигурацииПереопределяемый` в процедуре `ПриДобавленииПодсистем` предусмотрена ссылка на общий модуль, который содержит описание типовой конфигурации:

```
МодулиПодсистем.Добавить("ОбновлениеИнформационнойБазыБП");
```

В общем модуле `ОбновлениеИнформационнойБазыБП` в процедуре `ПриДобавленииПодсистемы` указано имя и версия типовой конфигурации, например:

```
Описание.Имя = "БухгалтерияПредприятияКОРП";
Описание.Версия= "3.0.38.31";
```

В этом случае можно смело изменить имя и синоним конфигурации на свои собственные, например `БухгалтерияПредприятияКОРП_CRM`, и ввести свою нумерацию версий конфигурации. Затем создать общий модуль вида `ОбновлениеИнформационнойБазы<Сокращение>`, например `ОбновлениеИнформационнойБазыCRM`, и подключить его в конце процедуры `ПриДобавленииПодсистем` общего модуля `ПодсистемыКонфигурацииПереопределяемый`:

```
МодулиПодсистем.Добавить("ОбновлениеИнформационнойБазыCRM");
```

В общем модуле `ОбновлениеИнформационнойБазыCRM` в процедуре `ПриДобавленииПодсистемы` указать имя доработанной конфигурации, версию и зависимость от типовой конфигурации, например:

```
Описание.Имя = "БухгалтерияПредприятияКОРП_CRM";
Описание.Версия = "1.0.1.1";
Описание.ТребуемыеПодсистемы.Добавить("БухгалтерияПредприятияКОРП");
```

Обработчики обновления для доработанной конфигурации следует размещать в этом же модуле и привязывать их к новой системе нумерации версий: «1.0.0.2» и т. д.

В дальнейшем при каждом обновлении доработанной конфигурации на новую версию типовой конфигурации необходимо будет увеличивать номер версии доработанной конфигурации, для того чтобы сработали все обработчики обновления.

Подробнее о настройке модулей `ПодсистемыКонфигурацииПереопределяемый` и `ОбновлениеИнформационнойБазы<Сокращение>` см. раздел Подготовка к использованию подсистемы.

Очистка устаревших данных

При разработке конфигураций периодически появляется необходимость заменять устаревшие объекты (справочники, документы и другие) на новые. С учетом стандарта Удаление устаревших объектов метаданных из конфигурации процедура состоит из следующих шагов:

- К имени устаревшего объекта добавляется префикс `Удалить` и создается обработчик обновления для переноса данных из устаревшего объекта в новый объект (версия 1).
- Позже, например через год, создается обработчик очистки устаревших данных, если они не удалялись сразу при переносе данных (версия 2).
- Затем выполняется удаление объектов метаданных из конфигурации (версия 3).

Версия 2 называется обязательной (или переходной). Рекомендуемая частота выпуска обязательных переходных версий описана в стандарте Обеспечение совместимости библиотек.

Переход на каждую версию должен быть последовательным:

- Переход на версию 1 требуется, чтобы перенести данные до их очистки (в версии 2) и до удаления объектов метаданных (в версии 3).
- Переход на версию 2 требуется, чтобы очистить регистры от записей, в измерениях которых есть ссылки на удаляемые объекты метаданных и предотвратить ошибку реструктуризации **Записи регистра стали неуникальными** при удалении ссылочных объектов метаданных (в версии 3).

Процесс удаления устаревших объектов в обязательных переходных версиях можно представить в виде конвейера, например, в первой версии происходит перенос данных устаревшего объекта 1 в новый объект 1, во второй версии очистка устаревшего объекта 1, а также перенос данных устаревшего объекта 2 в новый объект 2, в третьей версии удаление устаревшего объекта 1, очистка устаревшего объекта 2 и перенос данных устаревшего объекта 3 в новый объект 3. И так далее.

При разработке обработчиков очистки данных требуется учитывать все регистры, в которые включены устаревшие удаляемые ссылочные объекты, например, через определяемые типы библиотек, которые используются в измерениях регистров. Это требуется для предотвращения ошибки реструктуризации **Записи регистра стали неуникальными**.

Для упрощения очистки любых устаревших объектов и очистки рабочих регистров от записей, содержащих в измерениях значения устаревших типов объектов, предусмотрена процедура `ПриЗаполненииОбъектовПланируемыхКУдалению` общего модуля `ОбновлениеИнформационнойБазыПереопределяемый`. В этой процедуре указываются объекты с префиксом `Удалить`, например:

```
Процедура ПриЗаполненииОбъектовПланируемыхКУдалению(Объекты) Экспорт
    ОбновлениеИнформационнойБазы.ДобавитьОбъектПланируемыйКУдалению(Объекты,
        "Документ.УдалитьПроизвольныйЭД");
    ОбновлениеИнформационнойБазы.ДобавитьОбъектПланируемыйКУдалению(Объекты,
        "Перечисление.ХозяйственныеОперации.УдалитьСписаниеТоваровПереданныхПартнерам");
КонецПроцедуры
```

Чтобы сократить размер базы данных, можно очистить любую таблицу устаревшего объекта метаданных (до перехода на версию 3). Для этого следует указать третий параметр `Истина`:

```
    ОбновлениеИнформационнойБазы.ДобавитьОбъектПланируемыйКУдалению(Объекты,
        "Документ.УдалитьПроизвольныйЭД", Истина);
```

Однако следует учитывать, что для ссылочных объектов затратно делать удаление по одному объекту по сравнению с быстрой очисткой реструктуризации (при переходе на версию 3). Поэтому для ссылочных объектов этот признак ставить не рекомендуется.

Кроме устаревших объектов с префиксом `Удалить` можно указать лишние типы в измерениях регистров (например, полезно для библиотек). Записи в таких регистрах будут удалены при обнаружении в них ссылок указанных типов, например:

```
    ОбновлениеИнформационнойБазы.ДобавитьОбъектПланируемыйКУдалению(Объекты,
        "РегистрСведений.ВерсииОбъектов.Объект",
        Новый ОписаниеТипов("ДокументСсылка.Анкета"));
```

См. также пример в процедуре `ПриЗаполненииОбъектовПланируемыхКУдалению` общего модуля `УправлениеДоступомСлужебный`.

Следует помнить, что записи регистров, подчиненных регистратору, удаляются целиком для всего регистратора, если в измерении хотя бы одной из них нашлась ссылка указанного лишнего типа.

Процедура очистки выполняется автоматически в ходе отложенного обновления последним этапом после успешного завершения всех остальных отложенных обработчиков. При работе в модели сервиса устаревшие неразделенные данные очищаются с помощью регламентного задания **Очистка устаревших данных**, которое включается автоматически после обновления неразделенных данных и начинает очистку после успешного завершения отложенных обработчиков обновления во всех областях данных. Кроме того, администратор может запустить очистку вручную из формы **Очистка устаревших данных** (раздел **Администрирование** - **Обслуживание** - **Обновление программы**).

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, за исключением следующих:

- константа `КоличествоПотоковОбновленияИнформационнойБазы`,
- регистр сведений `ОбработчикиОбновленияОбщихДанных`,
- регистр сведений `ПотокиОбновления`.

В планах обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется отключать регистрацию изменений для следующих объектов метаданных подсистемы (см. также раздел Особенности создания начального образа подчиненного узла распределенной ИБ):

- константа `ДетализироватьОбновлениеИБВЖурналеРегистрации`,
- константа `СведенияОбОбновленииИБ`,
- константа `ОтложенноеОбновлениеЗавершеноУспешно`,
- константа `СведенияОБлокируемыхОбъектах`,
- регистр сведений `ВерсииПодсистем`,
- регистр сведений `ОбработчикиОбновления`,
- регистр сведений `ПрогрессОбновления`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется исключать следующие объекты метаданных:

- константа `ДетализироватьОбновлениеИБВЖурналеРегистрации`,
- константа `КоличествоПотоковОбновленияИнформационнойБазы`,
- константа `СведенияОбОбновленииИБ`,
- регистр сведений `ВерсииПодсистем`,
- регистр сведений `ОбработчикиОбновления`,
- регистр сведений `ОбработчикиОбновленияОбщихДанных`,
- регистр сведений `ПотокиОбновления`,
- регистр сведений `ПрогрессОбновления`.

Обновление конфигурации

Подсистема предназначена для автоматического обновления конфигурации информационной базы, а также установки исправлений ошибок (патчей) в режиме **1С:Предприятие** «по требованию» или в указанное время в будущем.

Обновление конфигурации не следует использовать, если в конфигурации включена возможность изменения объектов метаданных. В таких случаях рекомендуется выполнять обновление в режиме **Конфигуратор**.

Обновление конфигурации можно установить из основной конфигурации (в этом случае она будет применена к конфигурации базы данных) или из файла (в качестве файла может быть указан cfu-файл обновления конфигурации или cf-файл поставки конфигурации). При работе в файловом режиме возможна установка обновления «прямо сейчас» или при завершении работы. При работе в клиент-серверном режиме также возможно запланировать установку обновления на указанное время и выслать на почту отчет о результатах обновления.

Исправления (патчи) можно устанавливать как совместно с обновлением конфигурации, так и отдельно. В этом случае установка исправлений выполняется сразу, а внесенные изменения у всех активных пользователей применятся после перезапуска сеанса. Устаревшие патчи, которые уже неактуальны на новой версии конфигурации, автоматически удаляются при обновлении на новую версию.

При совместном внедрении с инструментарием **1С:Библиотека Интернет-поддержка пользователей** (подсистема **Получение обновлений программы**) также появляется возможность поиска и установки обновлений, а также исправлений ошибок (патчей) через Интернет.

Процесс установки обновления состоит из следующих шагов:

- Установка блокировки начала сеансов с информационной базой и завершение активных сеансов (в случае планирования обновления на указанное время выполняется непосредственно перед установкой обновления).
- Создание резервной копии (недоступно в клиент-серверном режиме работы).
- Загрузка конфигурации из файла (кроме случая, когда файл обновления уже загружен в основную конфигурацию).
- Обновление конфигурации базы данных.
- Установка исправлений (патчей).
- Выполнение обработчиков обновления.
- Сжатие таблиц информационной базы (только для файлового режима работы).
- Разрешение подключения новых соединений.
- Перезапуск сеанса (в случае если был выбран вариант установки обновления «прямо сейчас»).

> **Важно!**
>
> Подсистема недоступна в веб-клиенте.

Настройка

Для использования подсистемы в конфигурации необходимо разместить обработку `УстановкаОбновлений` и общую форму `УстановленныеИсправления` в командном интерфейсе.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к функциям подсистемы **Обновление конфигурации** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Выполнение обновления конфигурации информационной базы.

> **Примечание**
>
> Работа с подсистемой недоступна внешним пользователям (подробнее о внешних пользователях см. в разделе Пользователи).

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между приложениями. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Обсуждения

Подсистема предназначена для подключения и настройки интернет-сервиса **1С:Диалог или локального сервера системы Взаимодействия**. С ее помощью пользователи приложения смогут общаться друг с другом в режиме реального времени, создавать тематические обсуждения и вести переписку по конкретным документам (например, заказам, реализациям или контрагентам). Кроме того, к приложению возможно подключать чаты в мессенджере и социальной сети для общения с клиентами, контрагентами и другими внешними контактными лицами прямо в приложении. Подробнее см. документацию.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора приложения необходимо разместить:

- обработку `ПодключениеОбсуждений`;
- команды `ОтключениеОбсуждений` и `НастройкиСообщенийИзДругихПрограмм` обработки `ПодключениеОбсуждений`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к подсистеме следует использовать роль, приведенную ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Подключение и отключение обсуждений, настройка получения и отправки сообщений из мессенджеров и соц. сетей.

Для ведения тематических обсуждений не требуется дополнительных ролей. Для ведения контекстных обсуждений по конкретным объектам требуется роль для просмотра самих объектов.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных. Обсуждения хранятся в сервисе **1С:Диалог** или в локальном сервере системы взаимодействия.

Организации

Подсистема предоставляет унифицированный программный интерфейс для получения сводной информации по организациям, если они предусмотрены в конфигурации. Сам справочник организаций, сопутствующие таблицы и пользовательский интерфейс в состав подсистемы не входят, предполагается, что они уже разработаны в конфигурации. Подсистема предназначена для упрощения внедрения в конфигурацию различных (сторонних) библиотек и подсистем, которым также требуются сведения об организациях.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей дополнительных ролей не требуется, кроме тех, которые уже предусмотрены в конфигурации для работы с организациями.

Использование при разработке конфигурации

Для упрощения внедрения в конфигурацию различных (сторонних) библиотек и подсистем, которым также требуются сведения об организациях, следует:

- заменить все прямые обращения к справочнику организаций и сопутствующим таблицам на вызовы программного интерфейса общего модуля `ОрганизацииСервер`;
- заполнение этого программного интерфейса свойствами организаций и значениями выполнить в переопределяемом общем модуле `ОрганизацииПереопределяемый`.

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных. Сам справочник организаций и сопутствующие таблицы в состав подсистемы не входят, предполагается, что они уже разработаны в конфигурации и настройка обмена данными для них выполнена.

Отправка SMS

Подсистема **Отправка SMS** предназначена для отправки сообщений пользователям посредством SMS. Для отправки SMS потребуется заключение договора с любым поставщиком услуг, поддерживаемым подсистемой. Список поддерживаемых поставщиков услуг может быть расширен при внедрении.

Настройка

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора приложения необходимо разместить общую форму `НастройкаОтправкиSMS`. См. пример размещения в форме `Органайзер` обработки `ПанельАдминистрированияБСП` демонстрационной конфигурации.

Расширение списка поддерживаемых провайдеров SMS

Для расширения списка поддерживаемых провайдеров необходимо:

- Добавить провайдера в перечисление `ПровайдерыSMS`.
- В процедурах `ОтправитьSMS` и `СтатусДоставки` общего модуля `ОтправкаSMSПереопределяемый` и в процедуре `ПриПолученииАдресаПровайдераВИнтернете` общего модуля `ОтправкаSMSКлиентПереопределяемый` написать свою реализацию.

Пример:

```
Процедура ОтправитьSMS(ПараметрыОтправки, Результат) Экспорт
    Если ПараметрыОтправки.Провайдер = Перечисления.ПровайдерыSMS._ДемоДругойПровайдер Тогда
        // отправка смс через провайдера _ДемоДругойПровайдер
        // ...
    КонецЕсли;
КонецПроцедуры
Процедура СтатусДоставки(ИдентификаторСообщения, Провайдер, Логин, Пароль, Результат) Экспорт
    Если Провайдер = Перечисления.ПровайдерыSMS._ДемоДругойПровайдер Тогда
        // проверка статуса отправки смс через провайдера _ДемоДругойПровайдер
        // ...
    КонецЕсли;
КонецПроцедуры
Процедура ПриПолученииАдресаПровайдераВИнтернете(Провайдер, АдресВИнтернете) Экспорт

    Если Провайдер = ПредопределенноеЗначение("Перечисление.ПровайдерыSMS._ДемоДругойПровайдер") Тогда
        АдресВИнтернете = "http://yandex.ru/search/?text=рассылка%20смс";
    КонецЕсли;

КонецПроцедуры
```

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Отправка SMS, запрос статуса доставки.
2.	`ОтправкаSMS` Отправка SMS, запрос статуса доставки.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включить в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Отчет о движениях документа

Подсистема **Отчет о движениях документа** позволяет просматривать движения произвольных документов, которые при проведении отражают зафиксированные ими события в регистрах сведений, накопления, бухгалтерии и расчетов. Для ее использования в конфигурации также должны быть внедрены подсистемы **Варианты отчетов** и **Подключаемые команды**.

Настройка

Необходимо выбрать один из двух вариантов внедрения подсистемы: полный (рекомендуется) или выборочный.

В первом варианте достаточно выполнить внедрение подсистемы **Подключаемые команды** сразу ко всем документам конфигурации. В этом случае отчет **Движения документа** будет автоматически выведен в подменю **Отчеты** у тех документов, в метаданных которых заполнена коллекция **Движения** (но при этом он не выводится у документов, не формирующих движения).

При выборочном внедрении необходимо принять решение, в каких документах требуется просматривать их движения с помощью отчета. Рекомендуется подключать отчет ко всем документам, в метаданных которых заполнена коллекция **Движения**. Затем к выбранным документам подключить подсистему **Подключаемые команды** и определить список документов, в которых выводится отчет, с помощью вставки в процедуру `ПередДобавлениемКомандОтчетов` общего модуля `ВариантыОтчетовПереопределяемый`:

```
Процедура ПередДобавлениемКомандОтчетов(КомандыОтчетов, Параметры, СтандартнаяОбработка) Экспорт
    ДокументыСОтчетомОДвижениях = Новый Массив;
    ДокументыСОтчетомОДвижениях.Добавить(Метаданные.Документы._ДемоНачислениеЗарплаты);
    ДокументыСОтчетомОДвижениях.Добавить(Метаданные.Документы._ДемоПоступлениеТоваров);
    ДокументыСОтчетомОДвижениях.Добавить(Метаданные.Документы._ДемоРеализацияТоваров);
    Отчеты.ДвиженияДокумента.ДобавитьКомандуОтчетОДвиженияхДокумента(КомандыОтчетов, Параметры,     ДокументыСОтчетомОДвижениях);
КонецПроцедуры
```

Настройка прав доступа пользователей

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Просмотр отчета о движениях документа.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Оценка производительности

Подсистема **Оценка производительности** предназначена для оценки интегральной производительности системы по методике APDEX.

Настройка

Для настройки подсистемы **Оценка производительности** на сбор данных замеров производительности необходимо принять решение по поводу состава ключевых операций, время выполнения которых требуется замерять и отслеживать.

Ключевые операции – это такие операции, в процессе выполнения которых пользователь непрерывно ожидает отклика от информационной системы. Например: проведение некоторого документа, запись некоторого справочника, загрузка данных и т. п. Ключевые операции не могут включать в себя интерактивные действия пользователя.

Длительные ключевые операции – это такие ключевые операции, время выполнения которых зависит от количества обрабатываемых данных. Например, обработка нескольких документов: допустимо тратить на обработку одного документа некоторое время, однако на обработку 1000 документов времени уйдет больше, чем на обработку 10. В таком случае необходимо при выполнении ключевой операции ориентироваться не на время обработки всех документов, а на время обработки одного документа. В некоторых случаях в ходе длительной ключевой операции может выполняться несколько шагов, некоторые из которых зависят от объема обрабатываемых данных. В таком случае для каждого шага необходимо фиксировать время выполнения, чтобы при выполнении анализа легко определить, где именно возникла проблема.

Ключевые операции задаются по названию. Настройку приоритета и целевого времени ключевых операций должен будет выполнить администратор информационной базы, после того как в ИБ будут собраны замеры. Настройку приоритета и целевого времени ключевых операций выполнять не обязательно. Данная настройка необходима для последующего анализа замеров.

Для анализа замеров необходимо использовать отчет `ОценкаПроизводительности`. Поскольку ключевых операций может быть много, для целей анализа удобно объединять интересующие ключевые операции в группы с помощью справочника `ПрофилиКлючевыхОпераций`. Там же необходимо назначить целевое время и приоритет, которые будут учитываться при построении отчета.

Для упрощения подбора целевого времени ключевой операции можно воспользоваться автоматическим подбором целевого времени из формы списка или элемента справочника `КлючевыеОперации`. Автоматический подбор целевого времени решает обратную задачу: позволяет определить целевое время по известному значению APDEX. Для этого необходимо в форме подбора целевого времени указать ключевую операцию, период для получения данных и оценочный APDEX.

В коде конфигурации, в местах выполнения ключевых операций, надо вставить вызовы подсистемы **Оценка производительности** для замера времени их выполнения. При этом имя ключевой операции указывается в качестве параметра процедур начала и/или окончания замера времени.

Подсистема **Оценка производительности** поддерживает пять сценариев работы с замерами времени:

1. Начать замер на клиенте и автоматически завершить замер на клиенте. Кроме того, до момента завершения замера можно программно доопределить имя ключевой операции, наличие ошибки при выполнении и комментарий замера времени (если они еще не были известны на момент начала замера).
2. Начать замер на сервере и явно завершить его на сервере (по усмотрению разработчика).
3. Начать замер на клиенте и явно завершить его на клиенте (по усмотрению разработчика). Кроме того, до момента завершения замера можно программно доопределить имя ключевой операции, наличие ошибки при выполнении и комментарий замера времени (если они еще не были известны на момент начала замера).
4. Начать замер длительной ключевой операции на клиенте и явно завершить его на клиенте. При этом необходимо фиксировать выполнение каждого шага с указанием количества данных.
5. Начать замер длительной ключевой операции на сервере и явно завершить его на сервере. При этом необходимо фиксировать выполнение каждого шага с указанием количества данных.

Для выполнения замеров ключевой операции по сценариям № 1 и № 3 необходимо вызвать функцию `ЗамерВремени(Неопределено)` общего модуля `ОценкаПроизводительностиКлиент`, которая вернет уникальный идентификатор замера.

- Если параметр функции `АвтоЗавершение` = `Истина`, то замер времени завершится автоматически. Если же параметр функции `АвтоЗавершение` = `Ложь`, то для завершения замера времени необходимо вызвать процедуру `ЗавершитьЗамерВремени(УИДЗамера)` общего модуля `ОценкаПроизводительностиКлиент`, где `УИДЗамера` – уникальный идентификатор, который вернула функция `НачатьЗамерВремени`.
- Если параметр функции `ФиксироватьСОшибкой` = `Истина`, то замер времени будет записан с признаком `ВыполненСОшибкой` = `Истина`. Это особенно актуально, если `АвтоЗавершение` = `Истина` в случае замера времени проведения документа для автоматического разделения замеров на корректные и ошибочные, например, в случае некорректного заполнения документа. В конце такого замера необходимо явным образом снимать признак ошибки через явный вызов процедуры `ЗавершитьЗамерВремени(УИДЗамера, Ложь)` или `УстановитьПризнакОшибкиЗамера(УИДЗамера, Ложь)`.
- В случае если `КлючеваяОперация` = `Неопределено`, необходимо до автоматического завершения замера установить имя ключевой операции вызовом процедуры `УстановитьКлючевуюОперациюЗамера(УИДЗамера, КлючеваяОперация)` общего модуля `ОценкаПроизводительностиКлиент`.
- В случае если до завершения автозамера разработчик не установил имя ключевой операции, то данный замер при автозавершении будет удален из клиентского буфера и не будет записан в регистр сведений.

Для выполнения замеров ключевой операции по сценарию № 2 необходимо вызвать функцию `НачатьЗамерВремени` общего модуля `ОценкаПроизводительности`. Затем для завершения замера времени вызывать функцию `ЗакончитьЗамерВремени` общего модуля `ОценкаПроизводительности`.

Для выполнения замеров ключевой операции по сценариям № 4 и № 5 необходимо вызвать функцию `НачатьЗамерДлительнойОперации(ИмяКлючевойОперации)` общего модуля `ОценкаПроизводительностиКлиент` или `ОценкаПроизводительности` соответственно, которая вернет описание замера – соответствие, хранящее данные замера.

- Для фиксации шага необходимо вызвать процедуру `ЗафиксироватьЗамерДлительнойОперации(ОписаниеЗамера, КоличествоДанных, ИмяШага)`. Если вызвать `ЗафиксироватьЗамерДлительнойОперации` повторно с тем же именем шага, то данные суммируются с результатом предыдущей фиксации шага.
- Для завершения замера необходимо вызвать процедуру `ЗакончитьЗамерДлительнойОперации(ОписаниеЗамера, КоличествоДанных, ИмяШага)` – завершает замер и осуществляет запись коллекции замеров: полный, удельный и для каждого шага. При этом для удельного замера и замеров шагов операции время записывается как отношение времени, потраченного на выполнение шага, к количеству обработанных данных. Имена шагов формируются по принципу `ИмяКлючевойОперации.ИмяШага`.

Пример замера времени проведения документа на клиенте с разделением на замеры корректного проведения и возникновением исключения при проведении (сценарий № 1):

```
// Модуль формы документа _ДемоЗаказПокупателя
&НаКлиенте
Перем ИдентификаторЗамераПроведение, ИдентификаторЗамераПроведениеНеНужнаРегистрацияОшибки;
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
    Если ПараметрыЗаписи.РежимЗаписи = РежимЗаписиДокумента.Проведение Тогда
        ИдентификаторЗамераПроведение = ОценкаПроизводительностиКлиент.ЗамерВремени("_ДемоПроведениеДокумента");
        ИдентификаторЗамераПроведениеНеНужнаРегистрацияОшибки = ОценкаПроизводительностиКлиент.ЗамерВремени();
    КонецЕсли;
    // Далее следует текст обработчика формы -->>
КонецПроцедуры
&НаКлиенте
Процедура ПослеЗаписи(ПараметрыЗаписи)

    ОценкаПроизводительностиКлиент.УстановитьПризнакОшибкиЗамера(ИдентификаторЗамераПроведение, Ложь);

    ОценкаПроизводительностиКлиент.УстановитьКлючевуюОперациюЗамера(ИдентификаторЗамераПроведениеНеНужнаРегистрацияОшибки, "_ДемоПроведениеДоку

    // Далее следует текст обработчика формы -->>

КонецПроцедуры
```

Пример реализации сценария №1 можно найти в форме документа `_ДемоНачислениеЗарплаты` демонстрационной конфигурации.

Пример замера времени выполнения регламентного задания (сценарий № 2):

```
// Процедура обработки регламентного задания по выгрузке данных
Процедура ЭкспортОценкиПроизводительности(КаталогиЭкспорта) Экспорт

    ДатаНачала = ОценкаПроизводительности.НачатьЗамерВремени();
    // Далее следует текст выполнения регламентного задания -->>

    ОценкаПроизводительности.ЗакончитьЗамерВремени("ЭкспортОценкиПроизводительности", ДатаНачала);

КонецПроцедуры
```

Пример замера времени длительной операции на клиенте – формирование отчета в фоновом задании (сценарий № 3):

```
&НаКлиенте
Перем ИдентификаторЗамера, УникальныйИдентификаторФоновогоЗадания;
&НаКлиенте
Процедура СформироватьОтчет()
    ИдентификаторЗамера = ОценкаПроизводительностиКлиент.ЗамерВремени("ОтчетПоПродажам", Ложь, Ложь);
    // Далее следует текст запуска формирования отчета на сервере в фоновом задании -->>
КонецПроцедуры

&НаКлиенте
Процедура ПослеФормирования(Результат, ПараметрыФормирования) Экспорт
    // Здесь размещен код проверки результата фонового задания, формировавшего отчет -->>
    ОценкаПроизводительностиКлиент.ЗавершитьЗамерВремени(ИдентификаторЗамера);
КонецПроцедуры
```

Пример реализации сценария №3 смотрите в общей форме `ФормаОтчета` демонстрационной конфигурации.

Пример замера времени длительной операции на клиенте – создание элементов справочников в фоновом задании (сценарии № 4 и № 5):

```
Процедура ВыполнитьДействие(Параметры, АдресРезультата) Экспорт

    ОписаниеЗамера = ОценкаПроизводительности.НачатьЗамерДлительнойОперации("ДемоЗамерДлительнойОперации");
    КоличествоКонтрагентов = Параметры.КоличествоКонтрагентов;
    КоличествоБанковскихСчетовКонтрагента = Параметры.КоличествоБанковскихСчетовКонтрагента;
    УдалитьСозданные = Параметры.УдалитьСозданные;

    НаименованиеКонтрагента = НСтр("ru = 'Контрагент %1'");
    НаименованиеСчета = НСтр("ru = 'Банковский счет контрагента %1'");
    МассивОбъектов = Новый Массив;

    НачатьТранзакцию();
    Попытка
        Для СчетчикКонтрагента = 1 По КоличествоКонтрагентов Цикл
            КонтрагентОбъект = Справочники._ДемоКонтрагенты.СоздатьЭлемент();

            КонтрагентОбъект.Наименование = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НаименованиеКонтрагента, Формат(СчетчикКонт
            КонтрагентОбъект.Записать();
            ОценкаПроизводительности.ЗафиксироватьЗамерДлительнойОперации(ОписаниеЗамера, 1, "ЗаписьКонтрагента");
            МассивОбъектов.Добавить(КонтрагентОбъект.Ссылка);
            Для СчетчикСчета = 1 По КоличествоБанковскихСчетовКонтрагента Цикл
                СчетОбъект = Справочники._ДемоБанковскиеСчета.СоздатьЭлемент();
                СчетОбъект.Владелец = КонтрагентОбъект.Ссылка;
                СчетОбъект.Наименование = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(НаименованиеСчета, Формат(СчетчикКонтрагента,
                СчетОбъект.Записать();
                ОценкаПроизводительности.ЗафиксироватьЗамерДлительнойОперации(ОписаниеЗамера, 1, "ЗаписьБанковскогоСчета");
            КонецЦикла;
        КонецЦикла;
        ЗафиксироватьТранзакцию();
    Исключение
        ОтменитьТранзакцию();
    КонецПопытки;

    Если УдалитьСозданные Тогда
        Для Каждого Элемент Из МассивОбъектов Цикл
            ЭлементОбъект = Элемент.ПолучитьОбъект();
            ЭлементОбъект.Удалить();
        КонецЦикла;
    КонецЕсли;
    ОценкаПроизводительности.ЗакончитьЗамерДлительнойОперации(ОписаниеЗамера, МассивОбъектов.Количество(), "УдалениеКонтрагентов");

КонецПроцедуры
```

Пример реализации смотрите в модуле менеджера обработки `_ДемоЗамерДлительнойОперации` демонстрационной конфигурации.

При завершении работы пользователя часть выполненных клиентских замеров может быть потеряна. Связано это с тем, что клиентские замеры хранятся на клиенте, а на сервер передаются периодически с целью минимизации клиент-серверных вызовов. Периодичность задается константой `ОценкаПроизводительностиПериодЗаписи` (по умолчанию раз в минуту).

Размещение в командном интерфейсе

Необходимо разместить в командном интерфейсе ответственного за оценку производительности константу `ВыполнятьЗамерыПроизводительности` и обработку `ОценкаПроизводительности`. См. пример размещения в демонстрационной конфигурации в группе `Оценка производительности` формы `Обслуживание` обработки `ПанельАдминистрированияБСП`.

Для того чтобы система автоматически начала собирать замеры производительности, необходимо константе `ВыполнятьЗамерыПроизводительности` установить значение `Истина` в процедуре – обработчике обновления и первоначального заполнения ИБ.

Настройка прав доступа пользователей

Роли, используемые для работы с подсистемой:

№	Роли и их назначение
1.	`НастройкаИОценкаПроизводительности` Анализ результатов замера производительности ключевых операций в приложении.

Использование при разработке конфигурации

Настройка обмена данными

Все объекты метаданных подсистемы не должны включаться в планы обмена, предназначенные для синхронизации данных между различными программами, для организации распределенной информационной базы (РИБ) и автономного рабочего места. Как правило, конфигурация оборудования и требования к производительности в каждом узле различаются и замеры производительности ключевых операций хранятся независимо.

Однако в редких ситуациях, когда во всех узлах РИБ одинаковые конфигурации оборудования и требования к производительности, то в планы обмена следует включать все объекты подсистемы, содержащие данные.

Печать

Подсистема **Печать** предназначена для формирования печатных форм объектов на основе табличных макетов (формат MXL) и макетов офисных документов в формате Office Open XML (docx).

Подсистема предоставляет визуальный редактор табличных макетов печатных форм и макетов Office Open XML (DOCX). В ее состав входят также инструменты для размещения команд печати на формах в подменю **Печать**, предпросмотр печатных форм и различные сервисные возможности по сохранению печатных форм в файлы, отправке по электронной почте и формированию изображений QR-кодов. Кроме того, имеется программный интерфейс для формирования печатных форм по макету, встроенному в конфигурацию.

При наличии подсистемы **Мультиязычность** пользователю предоставляется возможность перевода макетов и формирования печатных форм на разных языках.

Настройка

Необходимо принять решение по поводу состава объектов конфигурации (справочников, документов и т. п.), которые требуется выводить на печать, и того, в каком виде должны формироваться печатные формы. Затем создать для них команды печати, разработав описательную часть, логику формирования печатной формы, и внести изменения в модули форм, в которых предполагается выводить команды печати.

Описательная часть находится в процедуре `ДобавитьКомандыПечати`, а логика формирования печатной формы зависит от того, в каком виде должны формироваться печатные формы:

- Формирование печатной формы в формате табличного документа (с предварительным просмотром или сразу на принтер).
- Формирование комплекта табличных документов (с предварительным просмотром или сразу на принтер).
- Формирование печатных форм с интерактивным запросом дополнительных параметров у пользователя.
- Вывод табличного документа в один из популярных форматов (Office Open XML, Microsoft Excel, Adobe PDF, HTML, текстовый документ и другие).
- Формирование печатной формы в виде офисных документов в формате Office Open XML (для тех случаев, когда возможностей табличного макета недостаточно).

Подключение объектов конфигурации

Необходимо принять решение, для каких справочников, документов и других объектов конфигурации разрабатываются команды печати и в каких их формах требуется выводить подменю с командами печати.

- Объекты конфигурации, являющиеся поставщиками команд печати, следует перечислить в процедуре `ПриОпределенииНастроекПечати` общего модуля `УправлениеПечатьюПереопределяемый`. Например:

```
Процедура ПриОпределенииНастроекПечати(Настройки) Экспорт
  Настройки.ОбъектыПечати.Добавить(Справочники._ДемоОрганизации);
  Настройки.ОбъектыПечати.Добавить(Документы._ДемоСчетНаОплатуПокупателю);
КонецПроцедуры
```

- В модуле менеджера каждого из этих объектов, в области `ПрограммныйИнтерфейс`, должны быть определены процедуры `ПриОпределенииНастроекПечати` и `ДобавитьКомандыПечати` по шаблону:

```
#Область ПрограммныйИнтерфейс
// Переопределяет настройки печати для объекта.
//
// Параметры:
//  Настройки - см. УправлениеПечатью.НастройкиПечатиОбъекта.
//
Процедура ПриОпределенииНастроекПечати(Настройки) Экспорт
       Настройки.ПриДобавленииКомандПечати = Истина;
КонецПроцедуры

// Заполняет список команд печати.
//
// Параметры:
//   КомандыПечати - см. УправлениеПечатью.СоздатьКоллекциюКомандПечати
//
Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт

КонецПроцедуры
#КонецОбласти
```

- Далее в формах указанных объектов, в которых требуется выводить подменю с командами печати, нужно:
  - Встроить подсистему Подключаемые команды.
  - Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд печати по шаблону:
  - Имя: `ПодменюПечать`.
  - Заголовок: `Печать`.
  - Вид: `Подменю`.
  - Отображение: `Картинка`.
  - Картинка: `Печать` (стандартная картинка). Если предполагается, что в этом подменю может быть выведено большое количество команд (более 10), то рекомендуется добавить вложенные группы кнопок с суффиксами `Важное`, `Обычное` и `СмТакже`. Например: `ПодменюПечатьВажное`, `ПодменюПечатьОбычное` и `ПодменюПечатьСмТакже`. Суффиксы этих групп указываются в свойстве `Важность` тех команд, которые должны выводиться в этих группах (подробнее про это свойство см. далее в таблице **Параметры команд печати**).

В формах журналов документов подменю **Печать** автоматически заполняется командами документов, входящих в журнал. В случаях, когда команды печати необходимо задавать непосредственно в модуле менеджера журнала документов, необходимо сделать соответствующую вставку кода в процедуре `ПриПолученииНастроекСпискаКомандПечати` общего модуля `УправлениеПечатьюПереопределяемый`:

```
// Дополнительные настройки списка команд печати.
//
// Параметры:
//  НастройкиСписка - Структура - модификаторы списка команд печати.
//   * МенеджерКомандПечати     - МенеджерОбъекта - менеджер объекта, в котором формируется список команд печати;
//   * АвтоматическоеЗаполнение - Булево - заполнять команды печати из объектов, входящих в состав журнала.
//                                         Значение по умолчанию: Истина.
//
Процедура ПриПолученииНастроекСпискаКомандПечати(НастройкиСписка) Экспорт

    Если НастройкиСписка.МенеджерКомандПечати = ЖурналыДокументов._ДемоСкладскиеДокументы Тогда
        НастройкиСписка.АвтоматическоеЗаполнение = Ложь;
    КонецЕсли;

КонецПроцедуры
```

Разработка команд печати

Затем необходимо разработать описательную часть, перечислив команды печати для каждого объекта в процедуре `ДобавитьКомандыПечати`, указав представление, идентификатор и другие параметры (подробнее см. таблицу ниже). Например:

```
// Счет на оплату
КомандаПечати = КомандыПечати.Добавить();
КомандаПечати.МенеджерПечати = "Документ._ДемоСчетНаОплатуПокупателю";
КомандаПечати.Идентификатор = "СчетЗаказ";
КомандаПечати.Представление = НСтр("ru = 'Счет на оплату'");
КомандаПечати.ПроверкаПроведенияПередПечатью = Истина;
```

Параметры команды печати:

Параметр	Тип	Описание
`Идентификатор` (обязательный)	`Строка` или `Массив`.	Идентификатор команды печати, по которому менеджер печати с которую необходимо сформировать. Пример: `КомандаПечати.Идентификатор = "СчетЗаказ";` Для печати нескольк указывать одновременно несколько их идентификаторов (строко массивом строк), например: `КомандаПечати.Идентификатор = "СчетЗаказ,ГарантийноеПисьмо";` Ес количество копий печати для печатной формы, то ее идентифика столько раз, сколько копий необходимо сформировать. При этом порядок следования печатных форм в комплекте будет соответст идентификаторов печатных форм, указанных в этом параметре. гарантийное письмо): `КомандаПечати.Идентификатор = "СчетЗаказ,СчетЗаказ,ГарантийноеПи` печатной формы может содержать в себе и альтернативный мен отличается от указанного в параметре `МенеджерПечати`, например `КомандаПечати.Идентификатор = "СчетЗаказ,Обработка._ДемоПечатная` этом примере `ГарантийноеПисьмо` формируется в менеджере печа `Обработка._ДемоПечатнаяФорма`, а `СчетЗаказ` – в менеджере печат `МенеджерПечати`. Например, "Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_ Для печатных форм, менеджером печати которых является общи в качестве идентификатора необходимо указать полный путь к м
`Представление` (обязательный)	`Строка`	Представление команды в меню `Печать`. Пример: `КомандаПечати.Представление = НСтр("ru = 'Счет на оплату'");`
`МенеджерПечати` (необязательный)	`Строка`	Имя объекта, в модуле менеджера которого располагается проце табличные документы для этой команды. Например: `КомандаПечати.МенеджерПечати = "Документ._ДемоСчетНаОплатуПокупа` Если печатная форма формируется автоматически по данным пе необходимо указать общий модуль `УправлениеПечатью`.
`ТипыОбъектовПечати` (необязательный)	`Массив`	Список типов объектов, для которых предназначена команда печ для команд печати в журналах документов, где требуется провер объекта перед вызовом менеджера печати. Если список не заполнен, то при автоматическом создании списк документов он заполняется типом объекта, из которого была имп

`Обработчик` (необязательный)	`Строка`	Клиентский обработчик команды, в который необходимо передат стандартного обработчика команды печать. Используется, напри формируется на клиенте. Формат `<ИмяОбщегоМодуля><ИмяПроцедуры>` используется, когда про модуле. Формат `<ИмяПроцедуры>` используется, когда процедура размеще отчета или обработки, указанной в `МенеджерПечати`. Пример: `КомандаПечати.Обработчик = "_ДемоСтандартныеПодсистемыКлиент.Печ` Пример обработчика в модуле формы: `// Формирует печатную форму <представление печатной формы>.`
		`//`
		`// Параметры:`
		`// ПараметрыПечати - Структура - Сведения о печатной форме.`
		`// * ОбъектыПечати - Массив - Массив ссылок выбранных объектов.`
		`// * Форма - ФормаКлиентскогоПриложения - Форма, из которой вызва`
		`// * ДополнительныеПараметры - Структура - Дополнительные парамет`
		`// Прочие ключи структуры соответствуют колонкам таблицы КомандыП`
		`// подробнее см. в функции УправлениеПечатью.СоздатьКоллекциюКома`
		`// &НаКлиенте Функция <ИмяФункции>(ПараметрыПечати) Экспорт`
		`// Обработчик печати. КонецФункции` Следует иметь в виду, что обработчик вызывается при помощи м качестве обработчика может выступать только функция. При это функции никак в дальнейшем не используется подсистемой.

`Порядок` (необязательный)	`Число`	Значение от 1 до 100, указывающее порядок размещения команд командам. Сортировка команд меню `Печать` осуществляется сна по представлению. Значение по умолчанию: 50.
`Картинка` (необязательный)	`Картинка`	Картинка, которая отображается возле команды в меню `Печать`. `КомандаПечати.Картинка = БиблиотекаКартинок.ФорматPDF;`

Параметр	Тип	Описание
`СписокФорм` (необязательный)	`Строка`	Имена форм, в которых должна отображаться команда, через зап указан, то команда печати будет отображаться во всех формах об подсистема `Печать`. Пример: `КомандаПечати.СписокФорм = "ФормаДокумента,ФормаСписка"`
`МестоРазмещения` (необязательный)	`Строка`	Имя командной панели формы, в которую необходимо разместит необходимо использовать только в случае, когда на форме разме подменю `Печать`. В остальных случаях место размещения необх формы при вызове процедуры `УправлениеПечатью.ПриСозданииНаСе`
`ЗаголовокФормы` (необязательный)	`Строка`	Произвольная строка, переопределяющая стандартных заголово **документов**. Пример: `КомандаПечати.ЗаголовокФормы = НСтр("ru = 'Настраиваемый комплек`
`ФункциональныеОпции` (необязательный)	`Строка`	Имена функциональных опций, от которых зависит доступность к запятую.
`УсловияВидимости` (необязательный)	`Массив`	Определяет видимость команды в зависимости от контекста. Для регистрации условий следует использовать процедуру `ПодключаемыеКоманды.ДобавитьУсловиеВидимостиКоманды()`. Условия объединяются по «И».
`ПроверкаПроведенияПередПечатью` (необязательный)	`Булево`	Признак необходимости проверки проведенности документов пер не указан, то проверка проведенности не выполняется. Пример: `КомандаПечати.ПроверкаПроведенияПередПечатью = Истина;`
`СразуНаПринтер` (необязательный)	`Булево`	Признак необходимости печати документов без предварительног принтер. Если параметр не указан, то при выборе команды печат предварительного просмотра **Печать документов**. Пример: `КомандаПечати.СразуНаПринтер = Истина;`
`ФорматСохранения` (необязательный)	`ТипФайлаТабличногоДокумента`	Применяется для быстрого сохранения печатной формы (без дог различные форматы, отличные от mxl. Если параметр не указан, mxl. Пример: `КомандаПечати.ФорматСохранения = ТипФайлаТабличногоДокумента.PDF` печати сразу открывается сформированный в формате pdf докум
`ПереопределитьПользовательскиеНастройкиКоличества` (необязательный)	`Булево`	Признак необходимости отключения в форме `ПечатьДокументов` м восстановления выбранного пользователем количества экземпля параметр не указан, то механизм сохранения/восстановления на открытии формы `ПечатьДокументов`. Пример: `КомандаПечати.ПереопределитьПользовательскиеНастройкиКоличества`
`ДополнитьКомплектВнешнимиПечатнымиФормами` (необязательный)	`Булево`	Признак необходимости дополнить комплект документов всеми г внешними печатными формами (подсистема `ДополнительныеОтчет` не указан, внешние печатные формы не добавляются в комплект `КомандаПечати.ДополнитьКомплектВнешнимиПечатнымиФормами = Истина`
`ФиксированныйКомплект` (необязательный)	`Булево`	Признак необходимости блокировки от изменения пользователем документов. Если параметр не указан, то пользователь сможет и печатные формы из комплекта в форме `ПечатьДокументов`, а так Пример: `КомандаПечати.ФиксированныйКомплект = Истина;`
`ДополнительныеПараметры` (необязательный)	`Структура`	Произвольные параметры для передачи в менеджер печати.
`НеВыполнятьЗаписьВФорме` (необязательный)	`Булево`	Признак необходимости отключения механизма записи объекта г печати. Используется в исключительных случаях. Если параметр запись объекта, в случае если в форме объекта установлен приз Пример: `КомандаПечати. НеВыполнятьЗаписьВФорме= Истина;`
`ТребуетсяРасширениеРаботыСФайлами` (необязательный)	`Булево`	Признак необходимости подключения расширения для работы с выполнением команды. Если параметр не указан, подключение р будет.

Доступна разработка печатных форм в форматах табличного документа (**mxl**) и офисного документа (**DOCX**). Разработка команд печати существенно различается в зависимости от того, в каком виде должна формироваться печатная форма:

- Для автоматической компоновки печатной формы по макету и данным объекта, см. раздел Разработка команды печати для автоматической компоновки печатной формы по макету и данным объекта. Данный способ компоновки следует использовать в качестве основного.
- Для формирования печатной формы в коде см. раздел Разработка процедуры «Печать». Данный способ формирования следует использовать для сложных печатных форм, для которых автоматический способ компоновки не подходит. Например в случае, когда печатная форма требует условный вывод колонок, иерархию, либо имеет сложную структуру. Для макетов таких печатных форм будут

действовать ограничения в части редактирования их пользователями. В частности, будет отсутствовать список доступных реквизитов, а также не будет возможности создания пользовательского макета путем копирования поставляемого.

- Для вывода табличного документа в один из популярных форматов (Office Open XML, Microsoft Excel, Adobe PDF, HTML, текстовый документ и другие) см. раздел Разработка процедуры «Печать».
- Для формирования печатных форм с интерактивным запросом дополнительных параметров у пользователя см. раздел Формирование печатной формы в клиентском контексте.

Разработка команды печати для автоматической компоновки печатной формы по макету и данным объекта

Минимальный набор параметров, необходимый для автоматической компоновки печатной формы - это имя макета, его представление в подменю **Печать** и модуль `УправлениеПечатью` в качестве менеджера печати:

```
Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт
  КомандаПечати = КомандыПечати.Добавить();
  КомандаПечати.МенеджерПечати = "УправлениеПечатью";
  КомандаПечати.Идентификатор = "Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_СчетНаОплату";
  КомандаПечати.Представление = НСтр("ru = 'Счет на оплату (на основе СКД)'");
КонецПроцедуры
```

Для печати помимо макетов в формате табличного документа (**mxl**) можно использовать макеты офисных документов (**DOCX**). При этом механизмы подготовки печатных форм унифицированы - принцип подготовки макетов единообразный, подготовка данных печати общая.

Подготовка данных печати для автоматической компоновки печатных форм

Общий принцип автоматической компоновки печатной формы состоит в следующем. Подсистема **Печать** берет созданный разработчиком или пользователем макет и путем последовательного перебора выполняет копирование строк макета в печатную форму, подставляя значения параметров. Источником данных при этом выступает схема компоновки данных, относящаяся к печатаемому объекту.

В простом случае схема содержит набор данных - запрос, выбирающий все реквизиты и табличные части объекта (ВЫБРАТЬ * ИЗ Таблица). При копировании строк макета учитываются параметры, указанные в строке. Если хотя бы один параметр ссылается на табличную часть, то строка копируется из макета в печатную форму столько раз, сколько строк в табличной части печатаемого объекта, при этом подставляются значения параметров из соответствующих строк.

Указанный способ формирования печатной формы позволяет программно получать список доступных полей объекта для использования в печатной форме, что дает возможность пользователю в режиме предприятия не только вносить косметические правки в существующие макеты печатных форм, но и создавать с нуля собственные печатные формы. Для разработчика упрощается процесс создания печатной формы, так как при таком способе формирования печатной формы требуется разработать только макет, а процедуру формирования печатной формы по макету разрабатывать не требуется.

Список полей объекта, доступных для использования в макете печатной формы в качестве параметров, извлекается из относящейся к объекту схемы компоновки данных. Таким образом, каждый объект может предоставлять для печати собственный набор полей, отличающийся от структуры объекта в метаданных. Это может быть полезно в случае, если часть полей объекта находится в связанных таблицах, например, в регистрах, а также, если объект содержит служебные реквизиты, которые необходимо скрыть.

Список полей является иерархическим, наподобие списка полей в настройках отчетов. То есть, в макете можно использовать параметры вроде `[Организация.Главный бухгалтер]`. Однако, способ получения подчиненных полей и их значений при компоновке печатной формы существенно отличается. При получении подчиненных полей не используется та же самая схема компоновки данных, что и для полей верхнего уровня. Для каждого поля определяется собственная схема компоновки данных, и список подчиненных полей вместе с данными зачитывается из нее. Таким образом любые поля, в том числе поля простых типов, могут содержать подчиненные поля, например, `[Контрагент.Наименование.Регистр символов.ВСЕ ПРОПИСНЫЕ]`. В этом примере поле `Наименование` строкового типа, но содержит подчиненную группу полей `Регистр символов`.

Благодаря возможности использования иерархии полей, для каждого объекта не требуется выносить в схему компоновки данных вложенные поля реквизитов, достаточно вывести только реквизиты объекта, остальные связанные поля могут быть получены по иерархии ("через точку").

Схема компоновки данных печати для каждого ссылочного объекта по умолчанию генерируется автоматически, но может быть определена явно в объекте метаданных в виде макета **ДанныеПечати**.

Например, в печатную форму счета на оплату необходимо вывести ФИО руководителя и главного бухгалтера. При этом документ **СчетНаОплату** в метаданных содержит только реквизит `Организация` (справочник), а сведения о руководителе и главном бухгалтере хранятся в регистре сведений **ОтветственныеЛицаОрганизаций**. В этом случае в справочнике **Организации** можно создать макет **ДанныеПечати**, помимо реквизитов справочника добавить в него поля `Руководитель` и `ГлавныйБухгалтер`, определить их логику заполнения. Таким образом, для макета документа **СчетНаОплату** поля руководителя и главного бухгалтера будут доступны через поле `Организация`: `[Организация.Руководитель]` и `[Организация.Главный бухгалтер]`.

При подготовке схемы компоновки данных в конфигураторе достаточно описать только набор данных. В качестве списка доступных полей в редакторе макета в режиме предприятия будет использована коллекция `ДоступныеПоляОтбора` из настроек компоновки данных.

> **Важно!** В списке полей СКД обязательно должно быть поле **Ссылка**. Именно по этому полю выполняется отбор данных. Если набор данных содержит таблицу, то также обязательно поле таблицы **НомерСтроки**.

В некоторых случаях удобнее не переопределять автоматический список полей объекта, а дополнять полями, хранящимися отдельно от объекта. Для этого нужно добавить в объекте метаданных макет со схемой компоновки данных по аналогии с макетом **ДанныеПечати**, но назвать **ДополнительныеДанныеПечати**. В этом случае автоматический список полей по прежнему будет генерироваться, но будет дополнен

из макета **ДополнительныеДанныеПечати**. При таком подходе новые реквизиты объекта не нужно будет каждый раз добавлять в схему компоновки данных, они будут добавляться автоматически. См. пример в макете `ДополнительныеДанныеПечати` справочника `_ДемоМестаХранения`.

Если в СКД используется набор данных **Объект**, то в модуле менеджера объекта необходимо разместить процедуру `ПриПодготовкеДанныхПечати`, в нее передается коллекция `ВнешниеНаборыДанных`, которая используется при компоновке данных, ее нужно заполнить.

Пример:

```
// Подготавливает данные печати.
//
// Параметры:
//  ИсточникиДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ИсточникиДанных
//  ВнешниеНаборыДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ВнешниеНаборыДанных
//  КодЯзыка - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.КодЯзыка
//  ДополнительныеПараметры - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ДополнительныеПараметры
//
Процедура ПриПодготовкеДанныхПечати(ИсточникиДанных, ВнешниеНаборыДанных, КодЯзыка, ДополнительныеПараметры) Экспорт
  ДанныеПечати = ДанныеПечати(ИсточникиДанных, КодЯзыка, ДополнительныеПараметры);
  ВнешниеНаборыДанных.Вставить("ДанныеПечати", ДанныеПечати);
КонецПроцедуры
```

Кроме этого, для полей ссылочных типов автоматически выводятся поля дополнительных реквизитов и сведений (при наличии подсистемы **Свойства**), а также поля контактной информации (при наличии подсистемы **Контактная информация**). Также к мультиязычным строковым реквизитам автоматически выводятся поля-представления на других языках (при наличии подсистемы **Мультиязычность**), для прочих строковых реквизитов - представления строк в разном регистре символов и т. д.

Для расширения состава вложенных полей предусмотрен программный интерфейс в виде процедур `ПриОпределенииИсточниковДанныхПечати` и `ПриПодготовкеДанныхПечати` общего модуля `УправлениеПечатьюПереопределяемый`.

Каждое поле объекта, даже нессылочного типа, при помощи механизма расширения состава полей может предоставлять свой набор подчиненных полей. Например поле **Наименование** справочника **ФизическиеЛица** может предоставлять подчиненное поле **ФамилияИнициалы**, данные которого физически не хранятся в информационной базе, а вычисляются "на лету" из наименования при помощи функции `ФизическиеЛицаКлиентСервер.ФамилияИнициалы`. В этом случае для описания набора подчиненных полей используется схема компоновки данных, в которой в качестве значений для обязательного поля **Ссылка** будут наименования (тип `Строка`). Примеры см. в процедурах `ПриОпределенииИсточниковДанныхПечати` и `ПриПодготовкеДанныхПечати` общего модуля `УправлениеПечатьюПереопределяемый`.

См. также примеры:

- в документе **_ДемоСчетНаОплатуПокупателю** Для табличного документа (**mxl**):
  - макет `ПФ_MXL_СчетНаОплату` - пример макета для автоматической компоновки печатной формы,
  - макет `ДанныеПечати` - пример использования набора данных типа **Запрос**; Для табличного документа (**DOCX**):
  - макет `ПФ_DOC_СчетНаОплатуСКД` - пример макета для автоматической компоновки печатной формы,
  - макет `ДанныеПечати` - пример использования набора данных типа **Запрос**;
- в справочнике **_ДемоОрганизации** макет ДанныеПечати - пример использования набора данных типа **Объект**.

Программное создание схемы компоновки данных печати

Схема компоновки данных печати может быть создана как в виде макета, так и программно из описания описания списка полей в виде таблицы значений или дерева значений. Для этого в модуле `УправлениеПечатью` предусмотрены функции `ТаблицаПолейДанныхПечати`, `ДеревоПолейДанныхПечати` и `СхемаКомпоновкиДанныхПечати`.

Например, в справочнике **Номенклатура** имеется реквизит **Штрихкод** с типом `Число`, но в печатной форме требуется выводить картинку, а не номер. Для этого программно создается СКД, описывающая поле `КартинкаШтрихкода`, и подключается в процедурах общего модуля `УправлениеПечатьюПереопределяемый` `ПриОпределенииИсточниковДанныхПечати` и `ПриПодготовкеДанныхПечати`:

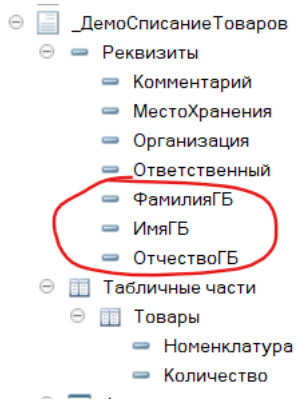
```
Процедура ПриОпределенииИсточниковДанныхПечати(Объект, ИсточникиДанныхПечати) Экспорт

  Если Объект = "Справочник.Номенклатура.Штрихкод" Тогда
    ИсточникиДанныхПечати.Добавить(СхемаДанныеПечатиШтрихкоды(), "Штрихкоды");
  КонецЕсли;

КонецПроцедуры

Процедура ПриПодготовкеДанныхПечати(ИсточникиДанных, ВнешниеНаборыДанных, ИдентификаторСхемыКомпоновкиДанных, КодЯзыка,

  Если ИдентификаторСхемыКомпоновкиДанных = "Штрихкоды" Тогда
    ВнешниеНаборыДанных.Вставить("Данные", ДанныеПечатиШтрихкоды(ИсточникиДанных));
  КонецЕсли;

КонецПроцедуры

Функция СхемаДанныеПечатиШтрихкоды()

  СписокПолей = УправлениеПечатью.ТаблицаПолейДанныхПечати();

  Поле = СписокПолей.Добавить();
  Поле.Идентификатор = "Ссылка";

  Поле = СписокПолей.Добавить();
  Поле.Идентификатор = "КартинкаШтрихкода";
  Поле.Представление = НСтр("ru = 'Картинка штрихкода'");
  Поле.Картинка = БиблиотекаКартинок.ТипКартинка;

  Возврат УправлениеПечатью.СхемаКомпоновкиДанныхПечати(СписокПолей);

КонецФункции

Функция ДанныеПечатиШтрихкоды(ИсточникиДанных)
  НаборДанных = Новый ТаблицаЗначений();
  НаборДанных.Колонки.Добавить("Ссылка");
  НаборДанных.Колонки.Добавить("КартинкаШтрихкода");

  Для Каждого ИсточникДанных Из ИсточникиДанных Цикл
    ЗначениеПолейДанных = НаборДанных.Добавить();
    ЗначениеПолейДанных.Ссылка = ИсточникДанных;

    ПараметрыШтрихкода = ГенерацияШтрихкода.ПараметрыГенерацииШтрихкода();
    ПараметрыШтрихкода.ТипКода = 1;
    ПараметрыШтрихкода.Ширина = 242;
    ПараметрыШтрихкода.Высота = 127;
    ПараметрыШтрихкода.Штрихкод = ИсточникДанных;

    ЗначениеПолейДанных.КартинкаШтрихкода = ГенерацияШтрихкода.ИзображениеШтрихкода(ПараметрыШтрихкода).Картинка;
  КонецЦикла;

  Возврат НаборДанных;

КонецФункции
```

Группировка полей объекта

При редактировании макета пользователь работает со списком доступных полей объекта. В случае, если полей много, для упрощения навигации в списке поля можно группировать. Группировка полей выполняется непосредственно в схеме компоновки данных печати при описании полей набора данных. Поскольку схема компоновки данных при печати используется только для получения данных, группы не могут использоваться в макете как поля, так как не содержат значений. Если требуется сгруппировать поля и при этом группу тоже использовать как поле, содержащее значение, то в набор данных необходимо добавить поле, установить для него значение простого типа и использовать его в качестве группировки.

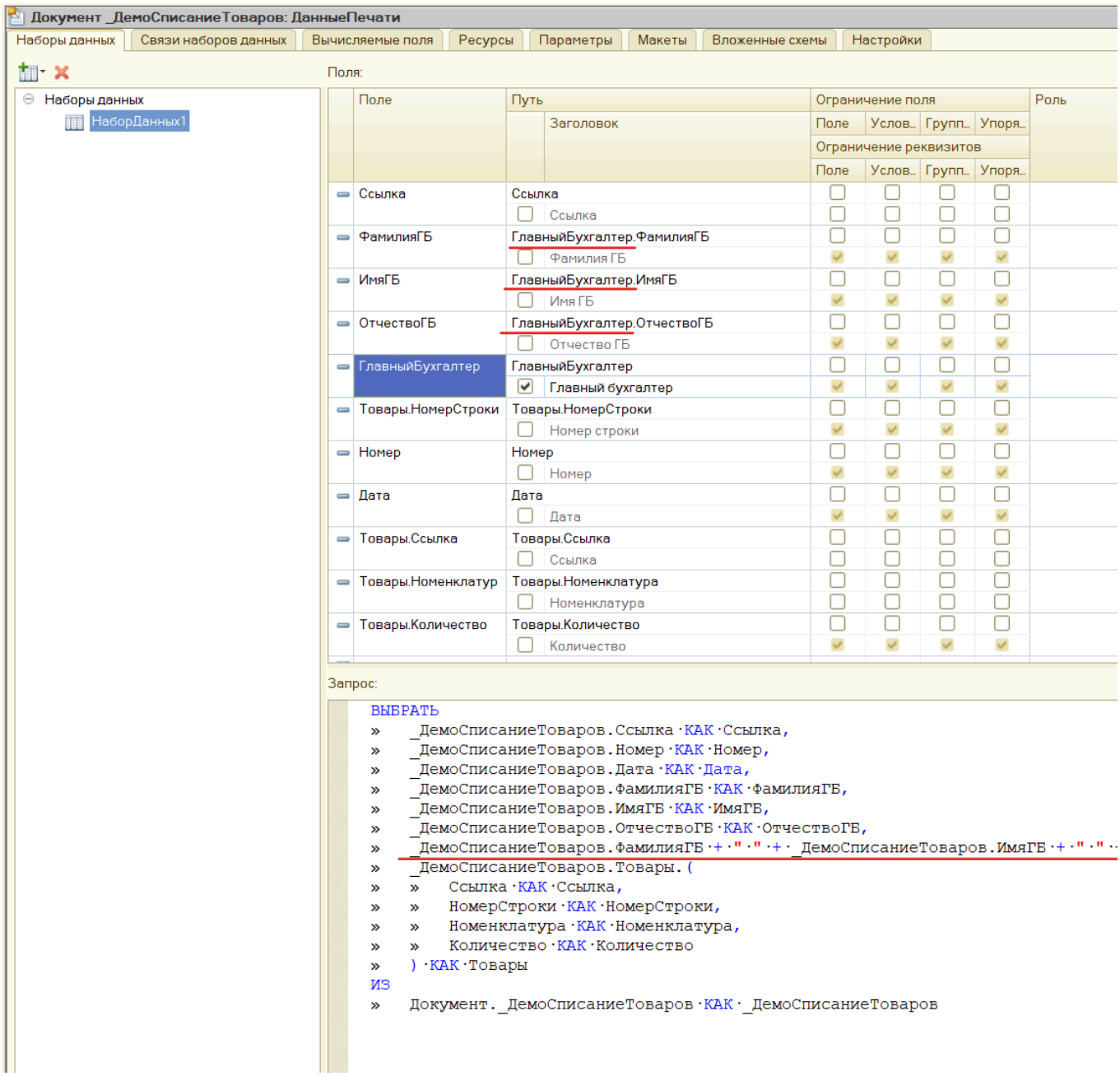
Например, в документе имеются поля:

- ФамилияГБ,
- ИмяГБ,
- ОтчествоГБ.

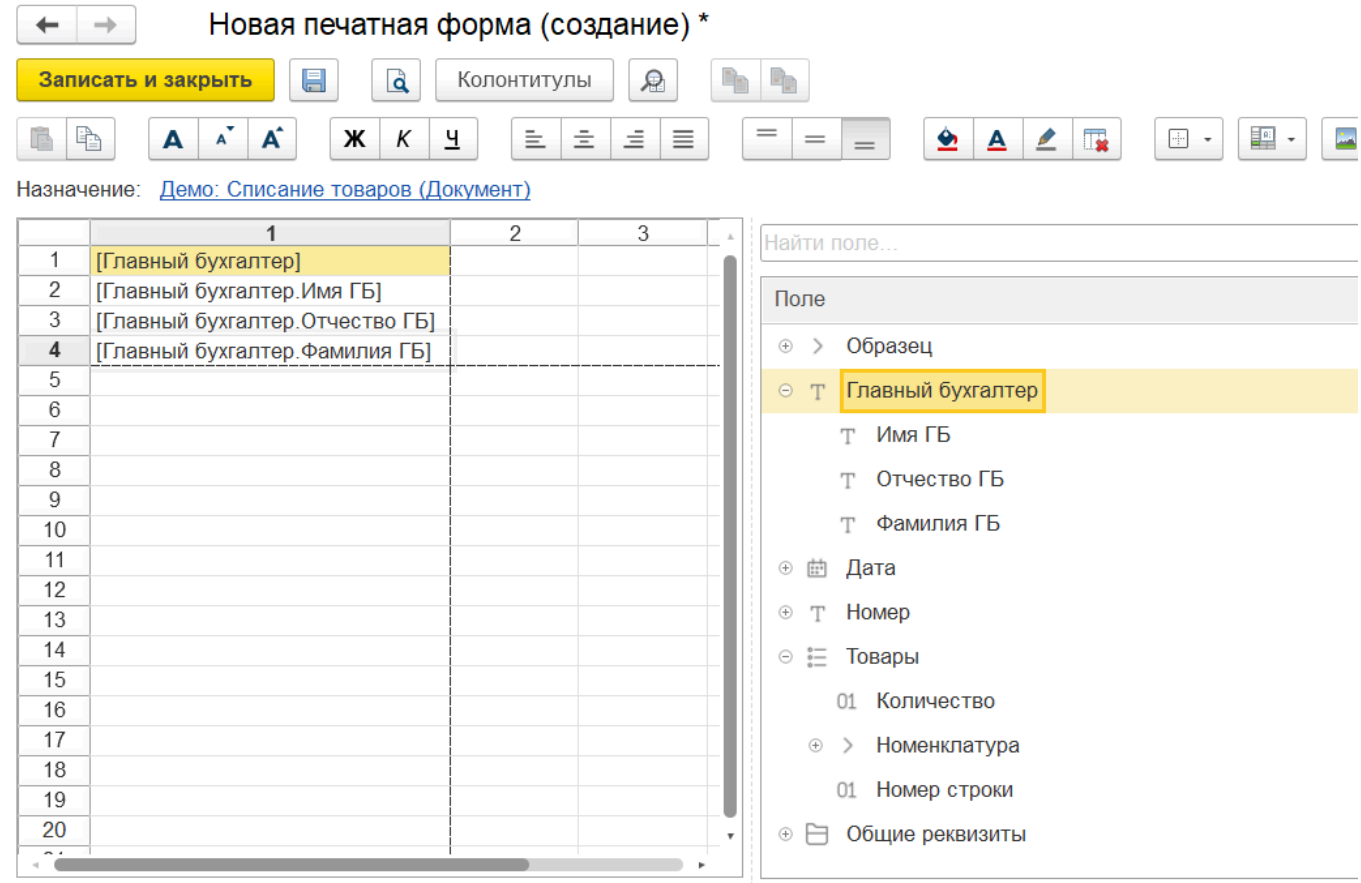


Требуется объединить их в группу **ГлавныйБухгалтер**, представление группы при выводе в печатную форму сделать путем конкатенации этих полей.

Для этого в СКД необходимо добавить поле `ГлавныйБухгалтер`, установить его значение в виде строки. У полей `ФамилияГБ`, `ИмяГБ`, `ОтчествоГБ` указать, что они подчинены полю `ГлавныйБухгалтер`.



В результате в списке доступных полей в редакторе макета поля ФИО будут находиться в группе `Главный бухгалтер`, при этом сама группа также может использоваться в макете.



Программно создание аналогичной схемы с группировкой и последующим заполнением данными можно описать следующим образом:

```
//  См. УправлениеПечатьюПереопределяемый.ПриОпределенииИсточниковДанныхПечати
Процедура ПриОпределенииИсточниковДанныхПечати(Объект, ИсточникиДанныхПечати) Экспорт

   СписокПолей = УправлениеПечатью.ДеревоПолейДанныхПечати();

   Поле = СписокПолей.Строки.Добавить();
   Поле.Идентификатор = "Ссылка";
   Поле.Представление = НСтр("ru = 'Ссылка'");
   Поле.ТипЗначения = Новый ОписаниеТипов();

   Группа = СписокПолей.Строки.Добавить();
   Группа.Идентификатор = "ГлавныйБухгалтер";
   Группа.Представление = НСтр("ru = 'Главный бухгалтер'");
   Группа.ТипЗначения = Новый ОписаниеТипов("Строка");

   Поле = Группа.Строки.Добавить();
   Поле.Идентификатор = "ФамилияГБ";
   Поле.Представление = НСтр("ru='Фамилия'");
   Поле.ТипЗначения = Новый ОписаниеТипов("Строка");

   Поле = Группа.Строки.Добавить();
   Поле.Идентификатор = "ИмяГБ";
   Поле.Представление = НСтр("ru='Имя'");
   Поле.ТипЗначения = Новый ОписаниеТипов("Строка");

   Поле = Группа.Строки.Добавить();
   Поле.Идентификатор = "ОтчествоГБ";
   Поле.Представление = НСтр("ru='Отчество'");
   Поле.ТипЗначения = Новый ОписаниеТипов("Строка");

   СхемаКомпоновкиДанныхПечати= УправлениеПечатью.СхемаКомпоновкиДанныхПечати(СписокПолей);
   ИсточникиДанныхПечати.Добавить(СхемаКомпоновкиДанныхПечати, "ПримерВыводаИерархии");

КонецПроцедуры

// Подготавливает данные печати.
//
// Параметры:
//  ИсточникиДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ИсточникиДанных
//  ВнешниеНаборыДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ВнешниеНаборыДанных
//  ИдентификаторСхемыКомпоновкиДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ИдентификаторСхемыКомпоновкиДанных
//  КодЯзыка - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.КодЯзыка
//  ДополнительныеПараметры - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ДополнительныеПараметры
//
Процедура ПриПодготовкеДанныхПечати(ИсточникиДанных, ВнешниеНаборыДанных, ИдентификаторСхемыКомпоновкиДанных, КодЯзыка,
   ДополнительныеПараметры) Экспорт

   Если ИдентификаторСхемыКомпоновкиДанных = "ПримерВыводаИерархии" Тогда

       НаборДанных = Новый ТаблицаЗначений();
       НаборДанных.Колонки.Добавить("Ссылка");
       НаборДанных.Колонки.Добавить("ГлавныйБухгалтер");
       НаборДанных.Колонки.Добавить("ФамилияГБ");
       НаборДанных.Колонки.Добавить("ИмяГБ");
       НаборДанных.Колонки.Добавить("ОтчествоГБ");

       Для Каждого ИсточникДанных Из ИсточникиДанных Цикл
           ЗначениеПолейДанных = НаборДанных.Добавить();
           ЗначениеПолейДанных.Ссылка = ИсточникДанных;
           ЗначениеПолейДанных.ГлавныйБухгалтер = ИсточникДанных.ФамилияГБ + " " + ИсточникДанных.ИмяГБ + " " + ИсточникДанных.ОтчествоГБ;
           ЗначениеПолейДанных.ФамилияГБ = ИсточникДанных.ФамилияГБ;
           ЗначениеПолейДанных.ИмяГБ = ИсточникДанных.ИмяГБ;
           ЗначениеПолейДанных.ОтчествоГБ = ИсточникДанных.ОтчествоГБ;
       КонецЦикла;

       ВнешниеНаборыДанных.Вставить("Данные", НаборДанных);

   КонецЕсли;

КонецПроцедуры
```

Добавление поля в существующую табличную часть

В некоторых случаях может потребоваться расширение уже имеющегося либо автогенерируемого списка полей табличной части объекта. Для этого достаточно добавить к объекту дополнительную схему компоновки данных с одноименной табличной частью, содержащей нужное поле плюс обязательное поле НомерСтроки. Пример программного добавления такой схемы - добавление поля `СуммаБезСкидки` в табличную часть `Товары` документа:

```
//  См. УправлениеПечатьюПереопределяемый.ПриОпределенииИсточниковДанныхПечати
Процедура ПриОпределенииИсточниковДанныхПечати(Объект, ИсточникиДанныхПечати) Экспорт

   СписокПолей = УправлениеПечатью.ДеревоПолейДанныхПечати();

   Поле = СписокПолей.Строки.Добавить();
   Поле.Идентификатор = "Ссылка"; // Обязательное поле
   Поле.Представление = НСтр("ru = 'Ссылка'");
   Поле.ТипЗначения = Новый ОписаниеТипов();

   ТаблицаТовары = СписокПолей.Строки.Добавить();
   ТаблицаТовары.Идентификатор = "Товары"; // Таблица, в которую добавляется поле
   ТаблицаТовары.Представление = НСтр("ru='Товары'");
   ТаблицаТовары.Картинка = БиблиотекаКартинок.ТипСписок;
   ТаблицаТовары.Таблица = Истина;
   ТаблицаТовары.ТипЗначения = Новый ОписаниеТипов("ТаблицаЗначений");

   ПолеТовары = ТаблицаТовары.Строки.Добавить();
   ПолеТовары.Идентификатор = "НомерСтроки"; // Обязательное поле
   ПолеТовары.Представление = НСтр("ru='Номер строки'");
   ПолеТовары.ТипЗначения = Новый ОписаниеТипов("Число");

   ПолеТовары = ТаблицаТовары.Строки.Добавить();
   ПолеТовары.Идентификатор = "СуммаБезСкидки"; // Добавляемое поле
   ПолеТовары.Представление = НСтр("ru='Сумма без скидки'");
   ПолеТовары.ТипЗначения = Новый ОписаниеТипов("Число");

   СхемаКомпоновкиДанныхПечати = УправлениеПечатью.СхемаКомпоновкиДанныхПечати(СписокПолей);
   ИсточникиДанныхПечати.Добавить(СхемаКомпоновкиДанныхПечати, "СуммаБезСкидкиСчетаНаОплату");

КонецПроцедуры

// Подготавливает данные печати.
//
// Параметры:
//  ИсточникиДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ИсточникиДанных
//  ВнешниеНаборыДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ВнешниеНаборыДанных
//  ИдентификаторСхемыКомпоновкиДанных - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ИдентификаторСхемыКомпоновкиДанных
//  КодЯзыка - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.КодЯзыка
//  ДополнительныеПараметры - см. УправлениеПечатьюПереопределяемый.ПриПодготовкеДанныхПечати.ДополнительныеПараметры
//
Процедура ПриПодготовкеДанныхПечати(ИсточникиДанных, ВнешниеНаборыДанных, ИдентификаторСхемыКомпоновкиДанных, КодЯзыка,
   ДополнительныеПараметры) Экспорт

   Если ИдентификаторСхемыКомпоновкиДанных = "СуммаБезСкидкиСчетаНаОплату" Тогда

      НаборДанных = Новый ТаблицаЗначений();
      НаборДанных.Колонки.Добавить("Ссылка");
      НаборДанных.Колонки.Добавить("Товары");

      Для Каждого ИсточникДанных Из ИсточникиДанных Цикл
         НаборДанныхТовары = Новый ТаблицаЗначений();
         НаборДанныхТовары.Колонки.Добавить("НомерСтроки", Новый ОписаниеТипов("Число"));
         НаборДанныхТовары.Колонки.Добавить("СуммаБезСкидки", Новый ОписаниеТипов("Число"));

         НоваяСтрока = НаборДанныхТовары.Добавить();
         НоваяСтрока.НомерСтроки = 1; // обязательно заполняем номер строки
         НоваяСтрока.СуммаБезСкидки = ...; // Заполнить согласно логике добавляемого поля.

         // ... дозаполнить остальные строки табличной части источника данных

         ЗначениеПолейДанных = НаборДанных.Добавить();
         ЗначениеПолейДанных.Ссылка = ИсточникДанных;
         ЗначениеПолейДанных.Товары = НаборДанныхТовары;
      КонецЦикла;

      ВнешниеНаборыДанных.Вставить("Данные", НаборДанных);

   КонецЕсли;

КонецПроцедуры
```

Добавление периодического реквизита в список доступных полей объекта

Для описания реквизита объекта, значение которого зависит от даты, необходимо создать отдельную схему компоновки данных с дополнительным полем **Период**. При вычислении значения реквизита учитывается наличие колонки **Период** и выбирается значение, соответствующее дате владельца реквизита.

Например, для справочника **МестаХранения** есть реквизит **Заведующий**, имеющий историю значений (периодический реквизит), хранящийся в отдельном регистре сведений **ЗаведующиеМестамиХранения**. Для того, чтобы добавить такой реквизит в список доступных для печати полей справочника **МестаХранения**, нужно создать отдельную схему компоновки данных (в виде макета или программно) с полями **Ссылка**,

Заведующий и **Период**. При получении заведующего местом хранения для какого-либо документа будет взято значение, актуальное на дату документа. См. пример в макете `ДополнительныеДанныеПечати` справочника `_ДемоМестаХранения`.

Особенности разработки макетов табличных документов (mxl) для автоматической компоновки печатной формы

Для автоматической компоновки печатной формы по макету и данным печати макет подготавливается особым способом.

- В макете не используется свойство `Заполнение` у ячеек, это значение для всех ячеек должно быть `Текст`.
- Тексты в ячейках для компоновщика печатной формы являются шаблонами, фрагменты текста, заключенные в квадратные скобки, считаются параметрами и заменяются данными печати.
- Именованные области строк в макете используются только для условного вывода этих строк, в остальных случаях области не используются.
- Именованные области колонок макета не используются.
- В качестве параметров выступают доступные поля объекта - владельца макета, допускается обращение к полям "через точку", например `[ДоговорКонтрагента.Дата]`.
- Если в тексте ячейки в качестве параметра указано поле табличной части объекта, то вся строка при компоновке печатной формы будет повторяться для каждой строки табличной части объекта.
- Для рисунка табличного документа в свойстве `Параметр расшифровки` путь к картинке в данных печати указывается в квадратных скобках, например `[Товары.Номенклатура.ФайлКартинки]`. При этом указанное поле должно быть ссылкой на присоединенный файл картинки (подсистема **Работа с файлами**).

Для подготовки макета рекомендуется использовать редактор макета в режиме предприятия. Подготовленный макет можно сохранить в файл (**Еще - Сохранить в файл...**), затем открыть в конфигураторе и скопировать содержимое в макет в метаданных.

Особенности разработки макетов офисных документов (DOCX) для автоматической компоновки печатной формы

Для автоматической компоновки печатной формы по макету и данным печати макет подготавливается особым способом.

- Фрагменты текста, заключенные в квадратные скобки, считаются параметрами и заменяются данными печати.
- В качестве параметров выступают доступные поля объекта - владельца макета, допускается обращение к полям "через точку", например `[ДоговорКонтрагента.Дата]`.
- Если в качестве параметра указано поле табличной части объекта, то вся область при компоновке печатной формы будет повторяться для каждой строки табличной части объекта.
- Для рисунка указывается путь к картинке в квадратных скобках, например `[Товары.Номенклатура.ФайлКартинки]`. При этом указанное поле должно быть ссылкой на присоединенный файл картинки (подсистема **Работа с файлами**).
- Для заполнения гиперссылки в поле адреса следует указать путь к адресу ссылки в квадратных скобках `[Товары.Номенклатура.ССылкаНаСтраницу]`
- Для обозначения вывода области по условию используются конструкции `{v8 Условие ТекстУсловия}`. Начало новой условной области является концом предыдущей. Окончание условной области `{/v8 Условие}`. `ТекстУсловия` - выражение, результат вычисления которого типа булево. Условия составляются на языке параметров.

Создание общих печатных форм

Создание общего макета для нескольких объектов не отличается от создания макета для конкретного объекта. Для возможности печати по общему макету необходимо наличие в данных печати у разных объектов одинаковых полей. Используемые в макете поля, которые отсутствуют в данных печати объекта, при подготовке печатной формы заменяются пустыми строками.

Общий макет необходимо разместить в обработке, обработку включить в состав подсистемы `ПодключаемыеОтчетыИОбработки`, в процедуре `ПриОпределенииНастроек` модуля менеджера обработки указать для каких объектов подключается команда печати и в процедуре `ДобавитьКомандыПечати` описать команду печати.

Пример подключения макета `Обработка._ДемоПечатьДоговораКуплиПродажи.ПФ_MXL_ДоговорКуплиПродажи` (см. демонстрационную конфигурацию):

```
// Определяет состав программного интерфейса для вызова из кода конфигурации.
//
// Параметры:
//  НастройкиПрограммногоИнтерфейса - Структура:
//    * ДобавитьКомандыПечати - Булево
//    * Размещение - Массив
//
Процедура ПриОпределенииНастроек(НастройкиПрограммногоИнтерфейса) Экспорт

  НастройкиПрограммногоИнтерфейса.Размещение.Добавить(Метаданные.Документы._ДемоСчетНаОплатуПокупателю);
  НастройкиПрограммногоИнтерфейса.Размещение.Добавить(Метаданные.Документы._ДемоРеализацияТоваров);

  НастройкиПрограммногоИнтерфейса.ДобавитьКомандыПечати = Истина;

КонецПроцедуры

// Заполняет список команд печати.
//
// Параметры:
//  КомандыПечати - см. УправлениеПечатью.СоздатьКоллекциюКомандПечати
//
Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт

  КомандаПечати = КомандыПечати.Добавить();
  КомандаПечати.МенеджерПечати = "УправлениеПечатью";
  КомандаПечати.Идентификатор = "Обработка._ДемоПечатьДоговораКуплиПродажи.ПФ_MXL_ДоговорКуплиПродажи";
  КомандаПечати.Представление = НСтр("ru = 'Договор купли-продажи (общий макет)'");

КонецПроцедуры
```

Разработка процедуры «Печать»

1. В модуль менеджера печати, указанного в параметре `МенеджерПечати`, добавить экспортную процедуру `Печать`:

   ```
   // Формирует печатные формы.
   //
   // Параметры:
   //  МассивОбъектов – Массив – ссылки на объекты, которые нужно распечатать;
   //  ПараметрыПечати – Структура – дополнительные настройки печати;
   //  КоллекцияПечатныхФорм – ТаблицаЗначений – сформированные табличные документы (выходной параметр)
   //  ОбъектыПечати – СписокЗначений – значение – ссылка на объект;
   //                                   представление – имя области, в которой был выведен объект (выходной параметр);
   //  ПараметрыВывода – Структура – дополнительные параметры сформированных табличных документов (выходной параметр).
   //
   Процедура Печать(МассивОбъектов, ПараметрыПечати, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт
   КонецПроцедуры
   ```

2. Разместить в конфигурации макет табличного документа согласно инструкциям раздела Размещение макетов печатных форм в конфигурации.
3. Добавить функцию, формирующую печатную форму (табличный документ), например:

   ```
   Функция ПечатьСчетаЗаказа(МассивОбъектов, ОбъектыПечати)

       . . .

       Возврат ТабличныйДокумент;
   КонецФункции
   ```

4. В процедуре `Печать` разместить код для идентификации требуемой печатной формы и вызова функции по ее формированию (созданную на шаге 3). Например:

   ```
   ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "СчетЗаказ");
   Если ПечатнаяФорма <> Неопределено Тогда
       ПечатнаяФорма.ТабличныйДокумент = ПечатьСчетаЗаказа(МассивОбъектов, ОбъектыПечати);
       ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
       ПечатнаяФорма.ПолныйПутьКМакету = "Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_СчетЗаказ";
   КонецЕсли;
   ```

Состав параметров процедуры `Печать` и их описание см. в процедуре `ПриПечати` общего модуля `УправлениеПечатьюПереопределяемый`.

Процедура `Печать` может формировать один или несколько табличных документов за один вызов. Процедура может быть реализована следующим образом:

- При необходимости устанавливаются `ПараметрыВывода`.
- Если формируется больше одного табличного документа, то рекомендуется получить необходимые данные до формирования табличных документов.
- При формировании табличного документа для нескольких объектов необходимо позаботиться о правильном порядке этих объектов в табличном документе. В процедуру `Печать`, в параметр `МассивОбъектов`, передается массив объектов в том порядке, в котором их выбрал

пользователь, однако «правильный» порядок вывода может быть иным – например, в документах, как правило, упорядочивание выполняется по полю `МоментВремени`.

- Если для каждого табличного документа данные получаются разными запросами, то желательно, чтобы документы упорядочивались одинаково.

В связи с тем, что макет может быть отредактирован пользователем в режиме **1С:Предприятие** и параметры в нем могут быть изменены или удалены, для повышения устойчивости кода формирования печатной формы рекомендуется избегать явного присвоения значений параметров в областях печати. Вместо этого следует использовать процедуру `ЗаполнитьЗначенияСвойств`.

Неправильно:

```
ОбластьПечати.Параметры.Организация = ДанныеПечати.Организация;
ОбластьПечати.Параметры.Контрагент = ДанныеПечати.Контрагент;
```

Правильно:

```
ЗаполнитьЗначенияСвойств(ОбластьПечати.Параметры, ДанныеПечати);
```

Пример реализации процедуры `Печать`:

```
Процедура Печать(МассивОбъектов, ПараметрыПечати, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт

    ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "СчетЗаказ");
    Если ПечатнаяФорма <> Неопределено Тогда
        ПечатнаяФорма.ТабличныйДокумент = ПечатьСчетаЗаказа(МассивОбъектов, ОбъектыПечати);
        ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
        ПечатнаяФорма.ПолныйПутьКМакету = "Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_СчетЗаказ";
    КонецЕсли;

КонецПроцедуры
```

Пример реализации функции, формирующей табличный документ:

```
Функция ПечатьСчетаЗаказа(МассивОбъектов, ОбъектыПечати)
    // Создаем табличный документ и устанавливаем имя параметров печати.
    ТабличныйДокумент = Новый ТабличныйДокумент;
    ТабличныйДокумент.КлючПараметровПечати = "ПараметрыПечати_СчетЗаказ";

    // Получаем запросом необходимые данные.
    Запрос = Новый Запрос();
    Запрос.Текст =
    "ВЫБРАТЬ
    |    СчетНаОплатуПокупателю.Ссылка КАК Ссылка,
    ......................
    |ГДЕ
    |    СчетНаОплатуПокупателю.Ссылка В(&МассивОбъектов)
    ......................
    |";
    Запрос.УстановитьПараметр("МассивОбъектов", МассивОбъектов);
    Шапка = Запрос.Выполнить().Выбрать();

    ПервыйДокумент = Истина;

    Пока Шапка.Следующий() Цикл
        Если Не ПервыйДокумент Тогда
            // Все документы нужно выводить на разных страницах.
            ТабличныйДокумент.ВывестиГоризонтальныйРазделительСтраниц();
        КонецЕсли;
        ПервыйДокумент = Ложь;
        // Запомним номер строки, с которой начали выводить текущий документ.
        НомерСтрокиНачало = ТабличныйДокумент.ВысотаТаблицы + 1;

        ......................

        // В табличном документе необходимо задать имя области, в которую был
        // выведен объект. Нужно для возможности печати комплектов документов.
        УправлениеПечатью.ЗадатьОбластьПечатиДокумента(ТабличныйДокумент,
            НомерСтрокиНачало, ОбъектыПечати, Шапка.Ссылка);
    КонецЦикла;
    Возврат ТабличныйДокумент;
КонецФункции
```

При разработке печатной формы можно предусмотреть ограничение ее вывода (на принтер, в буфер обмена, сохранение в файл). Для этого при формировании табличного документа необходимо установить у него соответствующее значение в свойстве `Вывод`, например:

```
ТабличныйДокумент.Вывод = ИспользованиеВывода.Запретить
```

Также имеется возможность размещения в печатной форме факсимильной подписи и печати. Для этого необходимо:

- в макете печатной формы добавить рисунок шириной 40 мм и соотношением сторон 1:1 для печати, 4:1 для подписи, и установить его свойства:
  - Имя: идентификатор вида `Подпись` или `Печать`, например, `ПодписьРуководителя`, `ПечатьОрганизации`;
  - Картинка: (не заполнено);
  - Размер картинки: `Пропорционально`;
  - Линия: `Точечная`.
- добавить код по получению картинки для добавленного рисунка в процедуре `ПриПолученииПодписейИПечатей` общего модуля `УправлениеПечатьюПереопределяемый`.

См. пример в макете `ПФ_MXL_СчетЗаказ` документа `_ДемоСчетНаОплатуПокупателю`.

Формирование печатной формы в клиентском контексте

В отдельных случаях для формирования некоторых печатных форм может потребоваться клиентский контекст. Например, для запроса дополнительных параметров печатной формы у пользователя непосредственно перед печатью. В таких случаях механизм формирования печатной формы необходимо размещать в клиентском модуле, а при описании команды печати в процедуре `ДобавитьКомандыПечати` использовать параметр `Обработчик` для передачи управления в этот модуль.

Принцип создания команд, использующих клиентский контекст, несколько отличается от основного. Процедура `Печать` модуля менеджера объекта не вызывается механизмами подсистемы, поэтому создавать ее для такой команды не требуется.

Процесс создания такой команды выглядит следующим образом:

1. В модуле менеджера объекта в процедуре `ДобавитьКомандыПечати` добавить описание команды (с использованием параметра `Обработчик`), например:

```
// Счет на оплату в Microsoft Word
КомандаПечати = КомандыПечати.Добавить();
КомандаПечати.МенеджерПечати = "Документ._ДемоСчетНаОплатуПокупателю";
КомандаПечати.Идентификатор = "СчетНаОплату(MSWord)";
КомандаПечати.Представление = НСтр("ru = 'Счет на оплату в Microsoft Word'");
КомандаПечати.Картинка = БиблиотекаКартинок.ФорматWord2007;
КомандаПечати.ПроверкаПроведенияПередПечатью = Истина;
КомандаПечати.Обработчик = "_ДемоСтандартныеПодсистемыКлиент.ПечатьСчетовНаОплатуПокупателю";
```

> **Примечание**
>
> Менеджер печати может быть использован для получения данных на печать (см. раздел Использование макетов в формате офисных документов при формировании печатных форм).

2. Добавить клиентскую экспортную функцию формирования печатной формы с единственным параметром, в который подсистема **Печать** будет передавать структуру параметров команды. Имя функции может быть произвольным, например:

```
Функция ПечатьСчетовНаОплатуПокупателю(ПараметрыПечати) Экспорт
    .
    .
    .
КонецФункции
```

> **Примечание 1**
>
> Из этой функции подсистема `Печать` не ожидает никакого результата. Использование функции вместо процедуры обусловлено тем, что ее вызов осуществляется при помощи метода `Вычислить`.
>
> **Примечание 2**
>
> Для вывода печатных форм на принтер пользователь должен обладать правом `Вывод`. При разработке клиентских команд печати следует учитывать наличие этого права.

Если клиентский контекст предполагает только запрос дополнительных параметров, а формирование табличного документа выполняется на сервере, то необходимо также выполнить инструкцию по разработке процедуры «Печать», а из клиентской функции выполнять передачу управления в процедуру `УправлениеПечатьюКлиент.ВыполнитьКомандуПечати`.

Вывод в печатную форму изображения QR-кода по заданной текстовой строке

В некоторых случаях в печатной форме документа полезно разместить изображение QR-кода, содержащего закодированные данные. Общий принцип встраивания функционала:

1. В макете печатной формы разместить рисунок типа `Картинка`.
2. В модуле менеджера печати вызвать процедуру `УправлениеПечатью.ДанныеQRКода`, передав в качестве параметра текстовую строку, которую необходимо закодировать с помощью QR-кода.
3. Полученный результат (двоичные данные) программно вывести в рисунок типа `Картинка`.

Пример реализации см. в демонстрационной конфигурации, в модуле менеджера документа `_ДемоСчетНаОплатуПокупателю` в процедуре `ПечатьКвитанции` и макете `ПФ_MXL_Квитанция`.

Прочие возможности

В процедуре `ПриОпределенииНастроекПечати` модуля менеджера объекта можно также указать другие обработчики подсистемы. При этом здесь же в модуле менеджера следует разместить соответствующую процедуру с реализацией обработчика:

Настройка	Название процедуры	Описание
`Настройки.ПриДобавленииКомандПечати = Истина;`	`ДобавитьКомандыПечати`	Описывает команды печати, предоставляемые объектов. См. Подключение объектов конфигурации
`Настройки.ПриОпределенииПолучателей = Истина;`	`ПриОпределенииПолучателей`	Позволяет определить короткий список получателей при отправке печатной формы по почте.

При отправке печатных форм по почте в некоторых случаях требуется предзаполнить письмо: указать список получателей, тему и текст письма. Для этого следует использовать процедуру менеджера `ПриОпределенииПолучателей`. Пример реализации см. в демонстрационной конфигурации, в модуле менеджера документа `_ДемоСчетНаОплатуПокупателю`. Шаблон для вставки:

```
// Добавляет сведения для отправки по электронной почте.
//
// Параметры:
//  ПараметрыОтправки - см. УправлениеПечатьюПереопределяемый.ПриПечати.ПараметрыВывода.ПараметрыОтправки
//  МассивОбъектов - см. УправлениеПечатьюПереопределяемый.ПриПечати.МассивОбъектов
//  КоллекцияПечатныхФорм - см. УправлениеПечатьюПереопределяемый.ПриПечати.КоллекцияПечатныхФорм
//
Процедура ПриОпределенииПолучателей(ПараметрыОтправки, МассивОбъектов, КоллекцияПечатныхФорм) Экспорт

КонецПроцедуры
```

Размещение макетов печатных форм в конфигурации

Макеты печатных форм рекомендуется размещать при объекте, в котором реализована логика формирования печатных форм. К имени макета следует добавить специальный префикс:

- макеты табличных документов – `ПФ_MXL`,
- макет Office Open XML – `ПФ_DOC`,

Например, `ПФ_DOC_СчетНаОплату`.

В процедурах формирования печатных форм для получения макета следует использовать процедуру `УправлениеПечатью.МакетПечатнойФормы (<Путь к макету>)`, где путь к макету может быть:

- для макетов в документах: `Документ.<Имя документа>.<Имя макета>`;
- для макетов в обработках: `Обработка.<Имя обработки>.<Имя макета>`;
- для общих макетов: `ОбщийМакет.<Имя макета>`.

Например:

```
УправлениеПечатью.МакетПечатнойФормы("Документ.СчетНаОплатуПокупателю.ПФ_MXL_СчетЗаказ");
```

> **Внимание!**
>
> При размещении макета печатной формы в обработке или отчете для его получения пользователь должен обладать правом Просмотр на соответствующий объект метаданных.

Вывод печатной формы на разных языках

Формирование печатных форм на разных языках полезно, например, если необходимо предоставить какие-либо документы иностранному контрагенту: прайс-лист, счет на оплату и т. д. Для этого пользователь добавляет требуемые языки в справочник `ЯзыкиПечатныхФорм`, самостоятельно переводит на них макет печатной формы и значения реквизитов объектов, выводимых в печатной форме, и затем при выводе печатной формы переключает ее на нужный язык. Эта возможность доступна при совместном внедрении с подсистемой **Мультиязычность** только для печатных форм с макетами табличного документа.

Принять решение, какие печатные формы требуется выводить на нескольких языках. Затем адаптировать эти печатные формы по следующей схеме:

1. Обеспечить ввод необходимых данных на нескольких языках.
2. Обеспечить перевод содержимого макета печатной формы на требуемые языки.
3. Обеспечить формирование печатной формы на указанном языке.

1. Обеспечить ввод необходимых данных на нескольких языках

Принять решение, строковые реквизиты каких объектов должны выводиться в печатную форму на разных языках. Например, это могут быть наименования товаров, контрагентов и т.п. Настроить для них мультиязычный ввод значений:

- в справочнике (документе) создать табличную часть `Представления`;
- добавить реквизит табличной части `КодЯзыка` (Строка, 10, Переменная);

- скопировать выводимые в печатную форму реквизиты в табличную часть `Представления`;
- в модуле формы объекта в процедуре `ПриСозданииНаСервере` добавить строку:

```
  МультиязычностьСервер.ПриСозданииНаСервере(ЭтотОбъект, Объект);
```

а также добавить в модуль формы процедуру:

```
&НаКлиенте
Процедура Подключаемый_Открытие(Элемент, СтандартнаяОбработка)
  МультиязычностьКлиент.ПриОткрытии(ЭтотОбъект, Объект, Элемент,  СтандартнаяОбработка);
КонецПроцедуры
```

См. пример в демонстрационной конфигурации в справочниках `_ДемоКонтрагенты` и `_ДемоНоменклатура`.

2. Обеспечить перевод содержимого макета печатной формы на требуемые языки.

Разработать макет печатной формы таким образом, чтобы статические заголовки, надписи и тексты размещались в макете табличного документа, а не заполнялись из кода (текст, формируемый в коде, недоступен для перевода на другой язык).

Например, неправильно:

- в макете в области `Заголовок` указан параметр `<ТекстЗаголовка>`,
- текст заголовка формируется в коде:

```
ТекстЗаголовка = СтрШаблон("Счет на оплату № %1 от %2", Номер, Дата);
ДанныеПечати.Вставить("ТекстЗаголовка", ТекстЗаголовка);
ОбластьМакета = Макет.ПолучитьОбласть("Заголовок");
ОбластьМакета.Параметры.Заполнить(ДанныеПечати);
ТабличныйДокумент.Вывести(ОбластьМакета);
```

Правильно:

- в макете в области `Заголовок` указать шаблон: `<Счет на оплату № [Номер] от [Дата]>`
- в коде подставлять параметры шаблона из данных:

```
ДанныеПечати.Вставить("Номер", Номер);
ДанныеПечати.Вставить("Дата", Дата);
ОбластьМакета = Макет.ПолучитьОбласть("Заголовок");
ОбластьМакета.Параметры.Заполнить(ДанныеПечати);
ТабличныйДокумент.Вывести(ОбластьМакета);
```

См. пример формирования печатной формы в процедуре `ПечатьСчетаЗаказа` модуля менеджера документа `_ДемоСчетНаОплатуПокупателю`.

3. Обеспечить формирование печатной формы на указанном языке

В процедуре `Печать` менеджера печати параметру `ДоступенВыводНаДругихЯзыках` элемента коллекции `КоллекцияПечатныхФорм` установить значение `Истина`. Например:

```
ПечатнаяФорма = УправлениеПечатью.СведенияОПечатнойФорме(КоллекцияПечатныхФорм, "Счет");
Если ПечатнаяФорма <> Неопределено Тогда
  ПечатнаяФорма.ТабличныйДокумент = ПечатьСчетаНаОплату(МассивОбъектов, ОбъектыПечати, ПараметрыВывода.КодЯзыка);
  ПечатнаяФорма.СинонимМакета = НСтр("ru = 'Счет на оплату'");
  ПечатнаяФорма.ПолныйПутьКМакету = "Документ.СчетНаОплатуПокупателю.ПФ_MXL_СчетНаОплату";
  ПечатнаяФорма.ДоступенВыводНаДругихЯзыках = Истина;
КонецЕсли;
```

При печати на другом языке код языка передается в параметре `ПараметрыВывода.КодЯзыка` процедуры `Печать` менеджера печати. Значение кода языка необходимо использовать при получении макета и данных, выводимых в табличный документ.

Получение макета печатной формы

Для получения макета на нужном языке в функцию `УправлениеПечатью.МакетПечатнойФормы` передать `КодЯзыка`:

```
Макет = УправлениеПечатью.МакетПечатнойФормы("Документ.СчетНаОплатуПокупателю.ПФ_MXL_СчетНаОплату", КодЯзыка);
```

Получение значений реквизитов на других языках

Получать значения реквизитов на требуемом языке можно с помощью запроса или через функции модуля `ОбщегоНазначения`: `ЗначениеРеквизитаОбъекта`, `ЗначенияРеквизитовОбъекта`, `ЗначениеРеквизитаОбъектов` и `ЗначенияРеквизитовОбъектов`.

> Вариант с запросом подходит не только для печатных форм, но и для любых запросов в отчетах, списках или других алгоритмов, которые выводят данные на требуемом языке.

В запросе добавить к исходной таблице левое соединение с табличной частью `Представления` (созданной на шаге 1) и передать в запрос `КодЯзыка`.

Например, было:

```
ВЫБРАТЬ
  ...
  СчетНаОплатуПокупателю.Контрагент КАК Контрагент,
  СчетНаОплатуПокупателю.Контрагент.НаименованиеДляПечати КАК КонтрагентНаименованиеДляПечати
  ...
ИЗ
  Документ.СчетНаОплатуПокупателю КАК СчетНаОплатуПокупателю
```

стало:

```
ВЫБРАТЬ
  ...
  СчетНаОплатуПокупателю.Контрагент КАК Контрагент,
  ЕстьNULL(КонтрагентыПредставления.НаименованиеДляПечати, СчетНаОплатуПокупателю.Контрагент.НаименованиеДляПечати) КАК КонтрагентНаименованиеД
  ...
ИЗ
  Документ.СчетНаОплатуПокупателю КАК СчетНаОплатуПокупателю
    ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Контрагенты.Представления КАК КонтрагентыПредставления
    ПО СчетНаОплатуПокупателю.Контрагент = КонтрагентыПредставления.Ссылка
    И (КонтрагентыПредставления.КодЯзыка = &КодЯзыка)
```

```
Запрос.УстановитьПараметр("КодЯзыка", КодЯзыка);
```

Также можно получать значения реквизитов на нужном языке программно с помощью функций `ЗначениеРеквизитаОбъекта`, `ЗначенияРеквизитовОбъекта`, `ЗначениеРеквизитаОбъектов`, `ЗначенияРеквизитовОбъектов`, передавая параметр `КодЯзыка`:

```
КонтрагентПредставление = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Шапка.Контрагент, "НаименованиеДляПечати", , КодЯзыка);
ДанныеПечати.Вставить("КонтрагентПредставление", ?(ЗначениеЗаполнено(КонтрагентПредставление), КонтрагентПредставление, Шапка.КонтрагентПредста
```

> В целях оптимизации производительности рекомендуется получать значения реквизитов на нужном языке в основном запросе данных печати.

Вывод чисел, дат и значений типа Булево

При выводе дат, чисел и значений типа Булево необходимо использовать функцию `Формат` с указанием языка. Например:

```
ДанныеПечати.Вставить("Дата", Формат(Шапка.Дата, "Л=" + КодЯзыка + "; ДЛФ=DD"));
```

```
ДанныеПечати.Вставить("Сумма", Формат(ТаблицаТовары.Итог("Сумма"), "Л=" + КодЯзыка));
```

```
ДанныеПечати.Вставить("ДоставкаКурьером", Формат(Шапка.ДоставкаКурьером, "Л=" + КодЯзыка));
```

Сумма прописью

Вывод суммы прописью доступен для языков, поддерживаемых платформой **1С:Предприятие** (см. описание функции ЧислоПрописью в синтаксис-помощнике). При совместном внедрении с подсистемой **Валюты** при вызове `РаботаСКурсамиВалют.СформироватьСуммуПрописью` указывать `КодЯзыка`:

```
ДанныеПечати.Вставить("СуммаПрописью",
  РаботаСКурсамиВалют.СформироватьСуммуПрописью(ДанныеПечати.ИтоговаяСумма, Шапка.ВалютаДокумента, , КодЯзыка));
```

При этом будут сформирован результат с учетом параметров прописи, введенных для валюты на указанном языке.

Контактная информация

Контактную информацию типа `Адрес` можно выводить в печатные формы транслитом. Для этого необходимо использовать функцию `КонтактнаяИнформация` общего модуля `УправлениеКонтактнойИнформацией`, устанавливая свойство `КодЯзыка`:

```
Отбор = УправлениеКонтактнойИнформацией.ОтборКонтактнойИнформации();
Отбор.ВидыКонтактнойИнформации.Добавить(УправлениеКонтактнойИнформацией.ВидКонтактнойИнформацииПоИмени("ЮридическийАдресОрганизации"));
Отбор.Дата = Дата;
Отбор.КодЯзыка = КодЯзыка;

ЮридическийАдрес = "";
КонтактнаяИнформация = УправлениеКонтактнойИнформацией.КонтактнаяИнформация(Организация, Отбор);
Если КонтактнаяИнформация.Количество() > 0 Тогда
  ЮридическийАдрес = КонтактнаяИнформация[0].Представление;
КонецЕсли;
```

Сведения об организации

При совместном внедрении с подсистемой **Организации** для получения значений реквизитов организации на другом языке также необходимо передавать параметр `КодЯзыка`:

```
СведенияОбОрганизации = ОрганизацииСервер.СведенияОбОрганизации(Шапка.Организация, , Шапка.Дата, КодЯзыка);
```

Для того предварительно в процедурах общего модуля `ОрганизацииПереопределяемый` необходимо реализовать получение сведений об организации с учетом языка, переданного в параметре `КодЯзыка`.

Свойства

При совместном внедрении с подсистемой **Свойства** пользователю доступно добавление произвольных реквизитов и сведений, которые он может разместить в макете печатной формы. Чтобы реквизиты и сведения, добавленные пользователем в макете, вывелись в печатной форме на требуемом языке, необходимо использовать функцию `ПредставленияЗначенийСвойств` общего модуля `УправлениеСвойствами`:

```
ПредставленияЗначенийСвойств = УправлениеСвойствами.ПредставленияЗначенийСвойств(МассивОбъектов, КодЯзыка);
Для Каждого Документ Из ДанныеПечати Цикл
  ...
  ОбластьМакета = Макет.ПолучитьОбласть(...);
  ОбластьМакета.Параметры.Заполнить(Документ);
  ОбластьМакета.Параметры.Заполнить(ПредставленияЗначенийСвойств[Документ.Ссылка]);

  ТабличныйДокумент.Вывести(ОбластьМакета);
  ...
КонецЦикла;
```

Пример формирования печатной формы на разных языках можно посмотреть в демонстрационной конфигурации в **Демо: Счете на оплату**, в печатной форме **Счет на оплату** и в процедуре `Печать` модуля менеджера документа `_ДемоСчетНаОплатуПокупателю`, печатная форма `Счет`.

Разработка печатных форм с использованием макетов в формате офисных документов (DOCX)

Помимо макетов в формате табличного документа (mxl), можно использовать макеты офисных документов (**DOCX**). При этом использование этих механизмов унифицировано.

Размещение в командном интерфейсе

Для редактирования макетов печатных форм следует разместить в командном интерфейсе ответственного за изменение макетов регистр сведений `ПользовательскиеМакетыПечати`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Печать** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`ИзменениеМакетовПечатныхФорм` Добавление и редактирование макетов печатных форм.
2.	`ВыводНаПринтерФайлБуферОбмена` (из подсистемы **Базовая функциональность**) Используется для разрешения вывода печатных форм на принтер и отправки их по электронной почте.
3.	`РедактированиеПечатныхФорм` Внесение изменений в сформированные печатные формы перед непосредственной отправкой на принтер, сохранением в файл или по электронной почте.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным.

№	Вспомогательные роли и их назначение
1.	`<ВыводПечатнойФормыНаПринтер>` Просмотр команды вывода печатных форм на принтер.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за поддержку макетов печатных форм	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ИзменениеМакетовПечатныхФорм`.
2.	Пользователь, которому разрешено получать и формировать печатные формы, а также выводить их на печать и пользоваться командами «быстрой печати» на принтер	• `БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ВыводНаПринтерФайлБуферОбмена` (из подсистемы **Базовая функциональность**), • `ВыводПечатнойФормыНаПринтер`.
3.	Пользователь, которому разрешено получать и формировать печатные формы, выводить их на печать, пользоваться командами «быстрой печати» на принтер, а также редактировать, сохранять и оправлять по почте печатные формы	• `БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ВыводНаПринтерФайлБуферОбмена` (из подсистемы **Базовая функциональность**), • `РедактированиеПечатныхФорм`, • `ВыводПечатнойФормыНаПринтер`.

Использование при разработке конфигурации

Перенос макета в другой объект

Обычно макет печатной формы располагается при объекте, но если макет является общим для двух и более объектов, то либо его следует располагать в обработке, либо это должен быть общий макет конфигурации. Таким образом, в процессе разработки конфигурации может возникнуть необходимость переместить макет из одного места конфигурации в другое. При этом следует помнить о том, что пользователь мог вносить изменения в макет, и эти изменения, хранящиеся в регистре сведений `ПользовательскиеМакетыПечати`, также необходимо переместить по новому адресу. Для этого необходимо написать отложенный обработчик обновления, в котором воспользоваться программным интерфейсом модуля `УправлениеПечатью`:

```
// Используется при переносе макета (объекта метаданных) печатной формы в другой объект.
// Предназначена для вызова в процедуре заполнения данных обновления (для "отложенного" обработчика).
// Регистрирует новый адрес макета для обработки.
//
// Параметры:
//  ИмяМакета   - Строка - Новое имя макета в формате
//                         "Документ.<ИмяДокумента>.<ИмяМакета>"
//                         "Обработка.<ИмяОбработки>.<ИмяМакета>"
//                         "ОбщийМакет.<ИмяМакета>":
//  Параметры - Структура - см. ОбновлениеИнформационнойБазы.ОсновныеПараметрыОтметкиКОбработке.
//
Процедура ЗарегистрироватьНовоеИмяМакета(ИмяМакета, Параметры) Экспорт
// Используется при переносе макета (объекта метаданных) печатной формы в другой объект.
// Предназначена для вызова в "отложенном" обработчике обновления.
// Переносит пользовательские данные, относящиеся макету на новый адрес.
//
// Параметры:
//  Макеты     - Соответствие - Сведения о прежних и новых именах макетов в формате
//                              "Документ.<ИмяДокумента>.<ИмяМакета>"
//                              "Обработка.<ИмяОбработки>.<ИмяМакета>"
//                              "ОбщийМакет.<ИмяМакета>":
//   * Ключ     - Строка - Новое имя макета.
//   * Значение - Строка - Прежнее имя макета.
//
//  Параметры - Структура - параметры, передаваемые в "отложенный" обработчик обновления.
//
Процедура ПеренестиПользовательскиеМакеты(Макеты, Параметры) Экспорт
```

Например, требуется перенести пользовательский макет (и переименовать) из `Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_СчетЗаказ` в `Обработка._ДемоПечатьЗаказов.ПФ_MXL_СчетУниверсальный`:

```
Процедура ПриДобавленииОбработчиковОбновления(Обработчики) Экспорт
    Обработчик = Обработчики.Добавить();
    Обработчик.Версия = "2.3.5.51";
    Обработчик.Процедура = "_ДемоОбновлениеИнформационнойБазыБСП.ПеренестиПользовательскиеМакетыВНовыеОбъекты";
    Обработчик.РежимВыполнения = "Отложенно";
    Обработчик.Комментарий = НСтр("ru = 'Изменение места хранения пользовательских макетов'");
    Обработчик.Идентификатор = Новый УникальныйИдентификатор("4c580abd-475c-4e93-a8e5-58219102f660");
    Обработчик.ПроцедураПроверки = "ОбновлениеИнформационнойБазы.ДанныеОбновленыНаНовуюВерсиюПрограммы";
    Обработчик.БлокируемыеОбъекты = "РегистрСведений.ПользовательскиеМакетыПечати";
    Обработчик.ПроцедураЗаполненияДанныхОбновления = "_ДемоОбновлениеИнформационнойБазыБСП.ЗарегистрироватьДанныеКОбработкеДляПереходаНаНовуюВе
    Обработчик.ЧитаемыеОбъекты = "РегистрСведений.ПользовательскиеМакетыПечати";
    Обработчик.ИзменяемыеОбъекты = "РегистрСведений.ПользовательскиеМакетыПечати";
КонецПроцедуры
Процедура ЗарегистрироватьДанныеКОбработкеДляПереходаНаНовуюВерсию(Параметры) Экспорт

    УправлениеПечатью.ЗарегистрироватьНовоеИмяМакета("Обработка._ДемоПечатьЗаказов.ПФ_MXL_СчетУниверсальный", Параметры);

КонецПроцедуры
Процедура ПеренестиПользовательскиеМакетыВНовыеОбъекты(Параметры) Экспорт

    Макеты = Новый Соответствие;
    Макеты.Вставить("Обработка._ДемоПечатьЗаказов.ПФ_MXL_СчетУниверсальный", "Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_СчетЗаказ");

    УправлениеПечатью.ПеренестиПользовательскиеМакеты(Макеты, Параметры);

КонецПроцедуры
```

Разработка команд печати в расширении конфигурации

Вместо поставки внешних печатных форм в виде внешних обработок рекомендуется вести их разработку с помощью расширений конфигурации. Для поставки команд печати внешних печатных форм в расширении конфигурации необходимо:

1. Добавить в расширение конфигурации обработку и включить ее в состав подсистемы `ПодключаемыеОтчетыИОбработки`.
2. В модуле менеджера обработки определить процедуру `ПриОпределенииНастроек` в соответствии с шаблоном подключения отчетов и обработок к другим объектам метаданных и реализовать команды печати:

```
#Область ПрограммныйИнтерфейс
// Определяет состав программного интерфейса для интеграции с конфигурацией.
//
// Параметры:
//   Настройки - Структура - Настройки интеграции этого объекта.
//       См. возвращаемое значение функции ПодключаемыеКоманды.НастройкиПодключаемыхОтчетовИОбработок().
//
Процедура ПриОпределенииНастроек(Настройки) Экспорт
    Настройки.Размещение.Добавить(Метаданные.Документы.ИмяДокумента);
    Настройки.ДобавитьКомандыПечати = Истина;
КонецПроцедуры
// Заполняет список команд печати.
//
// Параметры:
//   КомандыПечати - ТаблицаЗначений - Подробнее см. в УправлениеПечатью.СоздатьКоллекциюКомандПечати().
//
Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт
    // Код по добавлению команд печати.
КонецПроцедуры
#КонецОбласти
// Формирует печатные формы.
//
// Параметры:
//  МассивОбъектов – Массив – ссылки на объекты, которые нужно распечатать;
//  ПараметрыПечати – Структура – дополнительные настройки печати;
//  КоллекцияПечатныхФорм – ТаблицаЗначений – сформированные табличные документы (выходной параметр)
//  ОбъектыПечати – СписокЗначений – значение – ссылка на объект;
//                                      представление – имя области, в которой был выведен объект (выходной параметр);
//  ПараметрыВывода – Структура – дополнительные параметры сформированных табличных документов (выходной параметр).
//
Процедура Печать(МассивОбъектов, ПараметрыПечати, КоллекцияПечатныхФорм, ОбъектыПечати, ПараметрыВывода) Экспорт
КонецПроцедуры
```

Аналогичным образом в расширение конфигурации можно включить сразу несколько обработок с внешними печатными формами. Кроме того, предусмотрена возможность разрабатывать команды печати в отчетах. Например, это может потребоваться, если для формирования печатной формы используется система компоновки данных (СКД).

См. примеры в демонстрационной конфигурации: обработка `_ДемоПодключаемыеКомандыПечатьСчетовНаОплатуПокупателю` в составе расширения `_ДемоПодключаемыеКоманды` и обработка `_ДемоПечатьКарточкиОрганизации` в составе конфигурации.

При необходимости разработанные таким образом обработки также можно перенести без каких-либо доработок из расширения конфигурации в состав самой конфигурации.

> **Внимание!**
>
> При размещении макета печатной формы в обработке или отчете для его получения пользователь должен обладать правом `Просмотр` на соответствующий объект метаданных. Для этого необходимо в расширение добавить роль `ПолныеПрава` из конфигурации, а для неполноправных пользователей предусмотреть отдельную роль, либо также заимствовать имеющуюся в конфигурации, добавить в эти роли право `Просмотр` на обработку или отчет, в котором размещен макет.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места, кроме следующих:

- регистр сведений `ОбщиеПоставляемыеМакетыПечати`,
- регистр сведений `ПоставляемыеМакетыПечати`.

Подключаемые команды

Подсистема **Подключаемые команды** предоставляет программный интерфейс для вывода различных команд в формах, списках и журналах приложения. В частности, ее программный интерфейс используется для вывода команд создания на основании, заполнения объектов, а также команд подсистем Печать, Дополнительные отчеты и обработки и Варианты отчетов (подменю **Печать**, **Отчеты**, **Заполнить** и др.).

Настройка

Подключить формы объектов конфигурации

Для подключения форм справочников, документов и других объектов конфигурации, в которых требуется выводить подменю **Печать**, **Отчеты** или **Заполнить**, необходимо:

- В процедуре `ПриСозданииНаСервере` (обработчик события формы) вставить вызов по шаблону:

```
// СтандартныеПодсистемы.ПодключаемыеКоманды
ПодключаемыеКоманды.ПриСозданииНаСервере(ЭтотОбъект);
// Конец СтандартныеПодсистемы.ПодключаемыеКоманды
```

- В модуле формы вставить процедуры (обработчики команд):

```
// СтандартныеПодсистемы.ПодключаемыеКоманды
&НаКлиенте
Процедура Подключаемый_ВыполнитьКоманду(Команда)
    ПодключаемыеКомандыКлиент.НачатьВыполнениеКоманды(ЭтотОбъект, Команда, <ОбъектИлиТаблицаФормы>);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ПродолжитьВыполнениеКомандыНаСервере(ПараметрыВыполнения, ДополнительныеПараметры) Экспорт
    ВыполнитьКомандуНаСервере(ПараметрыВыполнения);
КонецПроцедуры
&НаСервере
Процедура ВыполнитьКомандуНаСервере(ПараметрыВыполнения)
    ПодключаемыеКоманды.ВыполнитьКоманду(ЭтотОбъект, ПараметрыВыполнения, <ОбъектИлиТаблицаФормы>);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ОбновитьКоманды()
    ПодключаемыеКомандыКлиентСервер.ОбновитьКоманды(ЭтотОбъект, <ОбъектИлиТаблицаФормы>);
КонецПроцедуры
// Конец СтандартныеПодсистемы.ПодключаемыеКоманды
```

- Если форма является формой объекта, тогда:
  - В параметре `ОбъектИлиТаблицаФормы` следует передавать реквизит формы типа `ДанныеФормыСтруктура`. Например:

```
ПодключаемыеКомандыКлиент.НачатьВыполнениеКоманды(ЭтотОбъект, Команда, Объект);
```

  - В обработчике `ПриЧтенииНаСервере` формы следует вставить вызов по шаблону:

```
&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)
    // СтандартныеПодсистемы.ПодключаемыеКоманды
    ПодключаемыеКомандыКлиентСервер.ОбновитьКоманды(ЭтотОбъект, <ОбъектФормы>);
    // Конец СтандартныеПодсистемы.ПодключаемыеКоманды
КонецПроцедуры
```

  - В обработчике `ПриОткрытии` формы следует вставить вызов по шаблону:

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
    // СтандартныеПодсистемы.ПодключаемыеКоманды
    ПодключаемыеКомандыКлиент.НачатьОбновлениеКоманд(ЭтотОбъект);
    // Конец СтандартныеПодсистемы.ПодключаемыеКоманды
КонецПроцедуры
```

- В обработчике `ПослеЗаписи` формы следует вставить вызов по шаблону:

```
&НаКлиенте
Процедура ПослеЗаписи(ПараметрыЗаписи)
    ПодключаемыеКомандыКлиент.ПослеЗаписи(ЭтотОбъект, <ОбъектФормы>, ПараметрыЗаписи);
КонецПроцедуры
```

- Также рекомендуется вызывать процедуры обновления видимости команд после изменения ключевых реквизитов (значения которых могут быть использованы в условиях видимости команд):
  - `ПодключаемыеКомандыКлиент.НачатьОбновлениеКоманд` – для обновления видимости команд на клиенте (подключает обработчик ожидания `Подключаемый_ОбновитьКоманды`);
  - `ПодключаемыеКомандыКлиентСервер.ОбновитьКоманды` – для обновления видимости команд на сервере (используется в том случае, если в процессе вызова уже делается серверный вызов).

- Если форма является формой списка, тогда:
  - В параметре `ОбъектИлиТаблицаФормы` следует передавать элемент формы типа `ТаблицаФормы`, связанный с динамическим списком. Например:

```
ПодключаемыеКомандыКлиент.ВыполнитьКоманду(ЭтотОбъект, Команда, Элементы.Список);
```

  - В обработчике `ПриАктивизацииСтроки` таблицы формы следует вставить вызов по шаблону:

```
&НаКлиенте
Процедура <ИмяТаблицы>ПриАктивизацииСтроки(Элемент)
    // СтандартныеПодсистемы.ПодключаемыеКоманды
    ПодключаемыеКомандыКлиент.НачатьОбновлениеКоманд(ЭтотОбъект);
    // Конец СтандартныеПодсистемы.ПодключаемыеКоманды
КонецПроцедуры
```

> **Внимание!**
>
> Поле `Ссылка` динамического списка формы должно быть доступно в обработчике команды. Для этого необходимо в свойствах поля `Ссылка`, вложенного в реквизит формы типа `ДинамическийСписок`, установить флажок **Использовать всегда**.

- Опционально. Добавить реквизит формы `ПараметрыПодключаемыхКоманд` произвольного типа. Это позволит не создавать реквизит формы динамически, что ускорит время открытия формы.

Размещение подключаемых команд от нескольких источников на форме

Допускается размещение на одной форме сразу нескольких наборов команд от разных источников. Это может потребоваться, например, при размещении на форме двух и более списков. В этом случае необходимо для каждого такого источника вызвать `ПодключаемыеКоманды.ПриСозданииНаСервере` и во втором параметре `ПараметрыРазмещения` указать этот источник в свойстве `ВладелецКоманд`. А в вызовах `ПодключаемыеКомандыКлиент.НачатьВыполнениеКоманды`, `ПодключаемыеКоманды.ВыполнитьКоманду` и `ПодключаемыеКомандыКлиентСервер.ОбновитьКоманды` не указывать параметр `Источник`.

Пример:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    ПараметрыРазмещения = ПодключаемыеКоманды.ПараметрыРазмещения();
    ПараметрыРазмещения.КомандныеПанель = Элементы.КомандыФормы;
    ПараметрыРазмещения.Источники = Метаданные.РегистрыСведений._ДемоРеестрСкладскихДокументов.Измерения.Ссылка.Тип;
    ПараметрыРазмещения.ВладелецКоманд = Элементы.Список;
    ПодключаемыеКоманды.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыРазмещения);

    ПараметрыРазмещения = ПодключаемыеКоманды.ПараметрыРазмещения();
    ПараметрыРазмещения.КомандныеПанель = Элементы.КомандныеПанельСписокРеализаций;
    ПараметрыРазмещения.Источники = Новый ОписаниеТипов("ДокументСсылка._ДемоРеализацияТоваров");
    ПараметрыРазмещения.ВладелецКоманд = Элементы.СписокРеализаций;
    ПодключаемыеКоманды.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыРазмещения);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ВыполнитьКоманду(Команда)
    ПодключаемыеКомандыКлиент.НачатьВыполнениеКоманды(ЭтотОбъект, Команда);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ПродолжитьВыполнениеКомандыНаСервере(ПараметрыВыполнения, ДополнительныеПараметры) Экспорт
    ВыполнитьКомандуНаСервере(ПараметрыВыполнения);
КонецПроцедуры

&НаСервере
Процедура ВыполнитьКомандуНаСервере(ПараметрыВыполнения)
    ПодключаемыеКоманды.ВыполнитьКоманду(ЭтотОбъект, ПараметрыВыполнения);
КонецПроцедуры

&НаКлиенте
Процедура Подключаемый_ОбновитьКоманды()
    ПодключаемыеКомандыКлиентСервер.ОбновитьКоманды(ЭтотОбъект);
КонецПроцедуры
```

См. пример в форме `РеестрСкладскихДокументов` регистра сведений `_ДемоРеестрСкладскихДокументов` в демонстрационной конфигурации.

Настройка прав доступа пользователей

Для настройки прав доступа дополнительных действий не требуется.

№	Роли и их назначение
1.	`БазовыеПраваБСП` или `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**): Просмотр информации.

Использование при разработке конфигурации

Подключение отчетов и обработок к механизмам конфигурации

В отчетах и обработках конфигурации и расширений можно разрабатывать дополнительные команды объектов конфигурации для вывода в подменю **Печать**, **Отчеты** или **Заполнить**, а также настраивать варианты отчета и переопределять поведение общей формы отчета. Для этого:

- Отчет или обработка должна входить в состав подсистемы `ПодключаемыеОтчетыИОбработки`.
- В модуле менеджера должна быть определена процедура `ПриОпределенииНастроек` по шаблону:

```
#Область ПрограммныйИнтерфейс
#Область ДляВызоваИзДругихПодсистем
// Определяет состав программного интерфейса для интеграции с конфигурацией.
//
// Параметры:
//   Настройки - Структура - Настройки интеграции этого объекта.
//       Описание см. в  ПодключаемыеКомандыПереопределяемый.ПриОпределенииСоставаНастроекПодключаемыхОбъектов.
//
Процедура ПриОпределенииНастроек(Настройки) Экспорт

КонецПроцедуры
#КонецОбласти
#КонецОбласти
```

- Структура параметра `Настройки` процедуры `ПриОпределенииНастроек` обработок, включенных в состав подсистемы `ПодключаемыеОтчетыИОбработки`:

Настройка	Тип	Описание
`Размещение`	`Массив` из `ОбъектМетаданных`	Объекты, к которым подключен этот объект. Процедуры `ДобавитьКомандыПечати`, `ДобавитьКомандыЗаполнения` и `ДобавитьКомандыОтчетов` вызываютс форм тех объектов метаданных, которые указаны в этом параметре.

`ДобавитьКомандыПечати`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Заполняет список команд печати.`
		`//`
		`// Параметры:`
		`// КомандыПечати - см. УправлениеПечатью.СоздатьКоллекциюКомандПечати.`
		`Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт`
		
		`КонецПроцедуры`

`ДобавитьКомандыЗаполнения`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Определяет список команд заполнения.`
		`//`
		`// Параметры:`
		`// КомандыЗаполнения - см. ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.Коман`
		`// Параметры - см. ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.Параметры.`
		`Процедура ДобавитьКомандыЗаполнения(КомандыЗаполнения, Параметры) Экспорт`
		
		`КонецПроцедуры`

- Структура параметра `Настройки` процедуры `ПриОпределенииНастроек` отчетов, включенных в состав подсистемы `ПодключаемыеОтчетыИОбработки`:

Настройка	Тип	Описание
`Размещение`	`Массив` из `ОбъектМетаданных`.	Объекты, к которым подключен этот объект. Процедуры `ДобавитьКомандыПечати`, `ДобавитьКомандыЗаполнения` и `ДобавитьКомандыОтчетов` вызывают форм тех объектов метаданных, которые указаны в этом параметре.

`ДобавитьКомандыПечати`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Заполняет список команд печати.`
		`//`
		`// Параметры:`
		`// КомандыПечати - см. УправлениеПечатью.СоздатьКоллекциюКомандПечати.`
		`Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт`
		
		`КонецПроцедуры`

`ДобавитьКомандыЗаполнения`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Определяет список команд заполнения.`
		`//`
		`// Параметры:`
		`// КомандыЗаполнения - см. ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.Кома`
		`// Параметры - см. ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.Параметры.`
		`Процедура ДобавитьКомандыЗаполнения(КомандыЗаполнения, Параметры) Экспорт`
		
		`КонецПроцедуры`

`ДобавитьКомандыОтчетов`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Определяет список команд отчетов.`
		`//`
		`// Параметры:`
		`// КомандыОтчетов - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.КомандыОтчето`
		`// Параметры - см. ВариантыОтчетовПереопределяемый.ПередДобавлениемКомандОтчетов.Параметры.`
		`Процедура ДобавитьКомандыОтчетов(КомандыОтчетов, Параметры) Экспорт`
		
		`КонецПроцедуры`

`НастроитьВариантыОтчета`	`Булево`	Если `Истина`, то в модуле менеджера следует определить процедуру по шаблону:
		`// Настройки размещения в панели отчетов.`
		`//`
		`// Параметры:`
		`// Настройки - см. ВариантыОтчетовПереопределяемый.НастроитьВариантыОтчетов.Настройки.`
		`// НастройкиОтчета - см. ВариантыОтчетов.ОписаниеОтчета.`
		`Процедура НастроитьВариантыОтчета(Настройки, НастройкиОтчета) Экспорт`
		
		`КонецПроцедуры`

`ОпределитьНастройкиФормы`	`Булево`	Отчет имеет программный интерфейс для тесной интеграции с формой отчета, в том числе может переопределять некоторые настройки формы и подписываться на ее события.
		Если `Истина` и отчет подключен к общей форме `ФормаОтчета`, тогда в модуле объекта отчета следу определить процедуру по шаблону:
		`// Настройки общей формы отчета подсистемы "Варианты отчетов".`
		`//`
		`// Параметры:`
		`// Форма - ФормаКлиентскогоПриложения, Неопределено`
		`// КлючВарианта - Строка, Неопределено`
		`// Настройки - см. ОтчетыКлиентСервер.НастройкиОтчетаПоУмолчанию.`
		`Процедура ОпределитьНастройкиФормы(Форма, КлючВарианта, Настройки) Экспорт`
		
		`КонецПроцедуры`

См. также примеры в демонстрационной базе: в объектах конфигурации и расширений, включенных в состав подсистемы `ПодключаемыеОтчетыИОбработки`.

Разработка и подключение команд создания на основании

Команды создания на основании предназначены для быстрого перехода к вводу нового документа (объекта) в тех случаях, когда для этого могут быть использованы данные уже существующего объекта. Например, документ может быть создан и предварительно заполнен на основании элемента справочника. Команды создания на основании определяются в коде конфигурации и расширений конфигурации и автоматически выводятся в подменю **Создать на основании** в формах элементов, документов, в списках и журналах объектов приложения.

Как и у стандартных команд ввода на основании, обработчиком команды является процедура `ОбработкаЗаполнения` в модуле объекта. Дополнительно подключаемые команды создания на основании предоставляют следующие возможности:

- автоматически формируемое подменю **Создать на основании** отображается в виде картинки, занимает меньше места, оставляя его для других важных команд;
- в конфигураторе не требуется настройка флажков видимости и настройка порядка команд в форме;
- управление составом, порядком и представлением команд со стороны объекта-основания;
- динамическое изменение состава команд в подменю в зависимости от выделенных элементов в списке и в зависимости от значений реквизитов объекта;
- группировка команд внутри подменю;

- использовать горячие клавиши и картинки для команд создания на основании.

Для исключения дублирования программно добавленных команд ввода на основании со стандартными (из метаданных) предусмотрен гибридный режим - генерируемые платформой команды ввода на основании на форме, при их наличии, автоматически переносятся в подменю команд, создаваемых программно. Однако, для упрощения поддержки конфигурации рекомендуется для всех объектов конфигурации использовать какой-то один из способов добавления команд ввода на основании (в метаданных или программный).

Порядок подключения:

1. Определить состав объектов, у которых есть команды создания на основании, а также объекты, которые могут являться основанием для создания других объектов. Перечислить все эти объекты в процедуре `ПриОпределенииОбъектовСКомандамиСозданияНаОсновании` общего модуля `СозданиеНаОснованииПереопределяемый`, пример:

```
Процедура ПриОпределенииОбъектовСКомандамиСозданияНаОсновании(Объекты) Экспорт
    Объекты.Добавить(Метаданные.Документы._ДемоСчетНаОплатуПокупателю);
    Объекты.Добавить(Метаданные.Документы._ ДемоРеализацияТоваров);
    Объекты.Добавить(Метаданные.Документы._ ДемоПеремещениеТоваров);
КонецПроцедуры
```

2. В модуле менеджера каждого из выбранных объектов вставить функцию `ДобавитьКомандуСоздатьНаОсновании` и процедуру `ДобавитьКомандыСозданияНаОсновании`.
3. В функции `ДобавитьКомандуСоздатьНаОсновании` реализовать добавление команды по созданию данного объекта. Пример для документа `_ДемоСписаниеТоваров`:

```
Функция ДобавитьКомандуСоздатьНаОсновании(КомандыСозданияНаОсновании) Экспорт
    Если ПравоДоступа("Добавление", Метаданные.Документы._ДемоСписаниеТоваров) Тогда
        КомандаСоздатьНаОсновании = КомандыСозданияНаОсновании.Добавить();
        КомандаСоздатьНаОсновании.Менеджер = Метаданные.Документы._ДемоСписаниеТоваров.ПолноеИмя();
        КомандаСоздатьНаОсновании.Представление = ОбщегоНазначения.ПредставлениеОбъекта(Метаданные.Документы._ДемоСписаниеТоваров);
        КомандаСоздатьНаОсновании.РежимЗаписи = "Проводить";
        Возврат КомандаСоздатьНаОсновании;
    КонецЕсли;
    Возврат Неопределено;
КонецФункции
```

   В простейшем случае можно использовать функцию `СозданиеНаОсновании.ДобавитьКомандуСозданияНаОсновании`, например:

```
Функция ДобавитьКомандуСоздатьНаОсновании(КомандыСозданияНаОсновании) Экспорт
    Возврат СозданиеНаОсновании.ДобавитьКомандуСозданияНаОсновании(КомандыСозданияНаОсновании, Метаданные.Документы._ДемоСписаниеТоваров);
КонецФункции
```

4. В процедуре `ДобавитьКомандыСозданияНаОсновании` вставить вызовы процедур добавления команд объектов, которые могут быть созданы на основании объекта, в котором размещена эта процедура, пример для документа `_ДемоСчетНаОплатуПокупателю`:

```
Процедура ДобавитьКомандыСозданияНаОсновании(КомандыСозданияНаОсновании, Параметры) Экспорт
    Документы._ДемоСписаниеТоваров.ДобавитьКомандуСоздатьНаОсновании(КомандыСозданияНаОсновании);
    Документы._ДемоРеализацияТоваров.ДобавитьКомандуСоздатьНаОсновании(КомандыСозданияНаОсновании);
    Документы._ДемоПеремещениеТоваров.ДобавитьКомандуСоздатьНаОсновании(КомандыСозданияНаОсновании);
КонецПроцедуры
```

Параметры команд Создания на основании

Параметр	Тип	Описание
`Представление` (обязательный)	`Строка`	Представление команды в форме. Пример: `Команда.Представление = НСтр("ru = 'Заполнить ТТН'");`
`Идентификатор` (необязательный)	`Строка`	Идентификатор команды, который используется для идентификации команды и определения имени команды в форме. Если идентификатор не указан, то имя команды будет определено автоматически.
`Важность` (необязательный)	`Строка`	Краткое имя (суффикс) группы в подменю, в которой следует вывести эту команду. Допустимо использовать: `Важное`, `Обычное` и `СмТакже`.
`Порядок` (необязательный)	`Число`	Значение от 1 до 100, указывающее порядок размещения команды по отношению к другим командам. Сортировка команд осуществляется сначала по полю `Порядок`, затем по представлению. Значение по умолчанию: 50.
`СочетаниеКлавиш` (необязательный)	`СочетаниеКлавиш`	Сочетание клавиш для быстрого вызова команды.

Параметры команд Создания на основании, настройки видимости и доступности команды

Параметр	Тип	Описание
`ТипПараметра` (необязательный)	`ОписаниеТипов`	Типы объектов, для которых предназначена эта команда. Используется, когда обработка (или другой поставщик команд) подключена к нескольким объектам конфигурации, для уточнения списка объектов, к которым подключена конкретная команда.
`ВидимостьВФормах` (необязательный)	`Строка`	Имена форм, в которых должна отображаться команда, через запятую. Используется для уточнения состава форм, к которым подключена конкретная команда. Если параметр не указан, то команда будет отображаться во всех формах. Пример: `Команда.ВидимостьВФормах = "ФормаДокумента"`
`ФункциональныеОпции` (необязательный)	`Строка`	Имена функциональных опций, определяющих видимость команды, через запятую
`УсловияВидимости` (необязательный)	`Массив`	Определяет видимость команды в зависимости от контекста. Для регистрации условий следует использовать процедуру `ПодключаемыеКоманды.ДобавитьУсловиеВидимостиКоманды()`. Условия объединяются по **«И»**.

Параметры команд Создания на основании, настройки выполнения команды

Параметр	Тип	Описание
`МножественныйВыбор` (необязательный)	`Булево`	Если `Истина`, то команда поддерживает множественный выбор и в 1 параметре обработчика команды будет передан массив ссылок. Если `Ложь`, то команда поддерживает только одиночный выбор и в 1 параметре обработчика команды будет передана 1 ссылка.
`РежимЗаписи` (необязательный)	`Строка`	Настройки дополнительных проверок и действий, связанных с записью объекта, выполняемых перед обработчиком команды: • `Записывать` – записывать новые и модифицированные объекты. • `Проводить` – проводить документы. Например: `Команда.РежимЗаписи = "Проводить";` Перед записью и проведением у пользователя запрашивается подтверждение. Значение по умолчанию: `Записывать`.

Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд создания на основании по шаблону:

- Имя: `ПодменюСоздатьНаОсновании`.
- Заголовок: **Создать на основании**.
- Вид: `Подменю`.
- Отображение: `Картинка`.
- Картинка: `ВводНаОсновании` (стандартная картинка).

Примеры команд и их обработчиков см. в документах `_ДемоПоступлениеТоваров`, `_ДемоРеализацияТоваров` и `_ДемоСписаниеТоваров`.

Разработка и подключение команд заполнения

Команды заполнения объектов предназначены для реализации различных способов заполнения реквизитов и табличных частей существующих документов (объектов). Например, для загрузки данных в документ из внешнего источника, для предварительного заполнения документа по шаблону и т.п. Список команд определяется в коде конфигурации и расширений конфигурации и автоматически группируется в подменю `Заполнить` в формах элементов и документов, в списках и журналах.

Порядок подключения:

1. Определить состав объектов, для которых требуется выводить команды заполнения, перечислить их в процедуре `ПриОпределенииОбъектовСКомандамиЗаполнения` общего модуля `ЗаполнениеОбъектовПереопределяемый`.
2. В модуле менеджера каждого из объектов, перечисленных в предыдущем шаге, вставить процедуру `ДобавитьКомандыЗаполнения` по шаблону:

```
// Определяет список команд заполнения.
//
// Параметры:
//   КомандыЗаполнения - ТаблицаЗначений - Таблица с командами заполнения. Для изменения.
//       См. описание 1 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//   Параметры - Структура - Вспомогательные параметры. Для чтения.
//       См. описание 2 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//
Процедура ДобавитьКомандыЗаполнения(КомандыЗаполнения, Параметры) Экспорт
    …
КонецПроцедуры
```

3. Команды заполнения объекта описать в процедуре `ДобавитьКомандыЗаполнения` модуля менеджера этого объекта, например:

```
// Определяет список команд заполнения.
//
// Параметры:
//   КомандыЗаполнения - ТаблицаЗначений - Таблица с командами заполнения. Для изменения.
//       См. описание 1 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//   Параметры - Структура - Вспомогательные параметры. Для чтения.
//       См. описание 2 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//
Процедура ДобавитьКомандыЗаполнения(КомандыЗаполнения, Параметры) Экспорт
    Команда = КомандыЗаполнения.Добавить();
    …
КонецПроцедуры
```

4. Команды заполнения, используемые сразу во всех (или во многих) объектах метаданных, описать в процедуре `ПередДобавлениемКомандЗаполнения` модуля `ЗаполнениеОбъектовПереопределяемый`, например:

```
Процедура ПередДобавлениемКомандЗаполнения(КомандыЗаполнения, НастройкиОбъекта, СтандартнаяОбработка) Экспорт
    ...
    // Универсальная команда заполнения для справочников.
    Если Метаданные.Справочники.Содержит(НастройкиОбъекта.Метаданные) Тогда
        Команда = КомандыЗаполнения.Добавить();
        …
    КонецЕсли;
    …
КонецПроцедуры
```

5. Команды заполнения, используемые для двух и более объектов описать в отдельной обработке, включить эту обработку в состав подсистемы `ПодключаемыеОтчетыИОбработки`, в модуле менеджера добавить процедуру `ПриОпределенииНастроек` по шаблону подключения отчетов и обработок к другим объектам метаданных, а также процедуру `ДобавитьКомандыЗаполнения`. Например:

```
#Область ПрограммныйИнтерфейс
// Определяет состав программного интерфейса для интеграции с конфигурацией.
//
// Параметры:
//   Настройки - Структура - Настройки интеграции этого объекта.
//       См. возвращаемое значение функции ПодключаемыеКоманды.НастройкиПодключаемыхОтчетовИОбработок.
//
Процедура ПриОпределенииНастроек(Настройки) Экспорт
    Настройки.Размещение.Добавить(Метаданные.Документы.ИмяДокумента);
    Настройки.ДобавитьКомандыЗаполнения = Истина;
КонецПроцедуры
// Определяет список команд заполнения.
//
// Параметры:
//   КомандыЗаполнения - ТаблицаЗначений - Таблица с командами заполнения. Для изменения.
//       См. описание 1 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//   Параметры - Структура - Вспомогательные параметры. Для чтения.
//       См. описание 2 параметра процедуры ЗаполнениеОбъектовПереопределяемый.ПередДобавлениемКомандЗаполнения.
//
Процедура ДобавитьКомандыЗаполнения(КомандыЗаполнения, Параметры) Экспорт
    Команда = КомандыЗаполнения.Добавить();
    …
КонецПроцедуры
#КонецОбласти
```

Процедура `ДобавитьКомандыЗаполнения` обработок, подключенных по такой технологии, вызывается для всех объектов метаданных, указанных в параметре `Размещение` процедуры `ПриОпределенииНастроек`.

Параметры команд Заполнения объектов

Параметр	Тип	Описание
`Представление` (обязательный)	`Строка`	Представление команды в форме. Пример: `Команда.Представление = НСтр("ru = 'Заполнить ТТН'");`
`Идентификатор` (необязательный)	`Строка`	Идентификатор команды, который используется для идентификации команды и определения имени команды в форме. Если идентификатор не указан, то имя команды будет определено автоматически.
`Важность` (необязательный)	`Строка`	Краткое имя (суффикс) группы в подменю, в которой следует вывести эту команду. Допустимо использовать: `Важное`, `Обычное` и `СмТакже`.
`Порядок` (необязательный)	`Число`	Значение от 1 до 100, указывающее порядок размещения команды по отношению к другим командам. Сортировка команд осуществляется сначала по полю `Порядок`, затем по представлению. Значение по умолчанию: 50.
`СочетаниеКлавиш` (необязательный)	`СочетаниеКлавиш`	Сочетание клавиш для быстрого вызова команды.

Параметры команд Заполнения объектов, обработчик команды заполнения

Параметр	Тип	Описание
`Менеджер` (необязательный)	`Строка`	Полное имя объекта метаданных, отвечающего за выполнение команды. Заполняется автоматически полным именем объекта, в модуле менеджера которого добавлена команда.

`Обработчик` (обязательный, если не указано `ИмяФормы`)	`Строка`	Имя процедуры, обрабатывающей основное действие команды.
		Пример № 1 – обработчик размещен в общем модуле:
		`Команда.Обработчик = "ПродажиКлиент.ВвестиСчетФактуру";`
		Пример № 2 – обработчик размещен в модуле формы или менеджера:
		`Команда.Обработчик = "ЗаполнитьТТН";`
		Если `ИмяФормы` заполнено, то в модуле указанной формы ожидается клиентская процедура по шаблону:
		`&НаКлиенте`
		`Процедура <ИмяПроцедуры>(<ИмяПараметраКоманды>, ПараметрыВыполнения) Экспорт`
		`// Код обработчика команды.`
		`КонецПроцедуры`
		Если `ИмяФормы` не заполнено, то в модуле менеджера объекта, указанного в `Менеджер`, ожидается серверная процедура по шаблону:
		`Процедура <ИмяПроцедуры>(<ИмяПараметраКоманды>, ПараметрыВыполнения) Экспорт`
		`// Код обработчика команды.`
		`КонецПроцедуры`
		Имя 1-го параметра обработчика команды `ИмяПараметраКоманды` рекомендуется задавать в соответствии с типами объектов, для которых предназначена команда. Также следует учитывать, что тип этого параметра зависит от настройки `МножественныйВыбор`. Если `МножественныйВыбор` = `Истина`, то передается массив ссылок, в противном случае передается ссылка. Например, если команда предназначена для заполнения документов `СчетаНаОплату` и `МножественныйВыбор` включен, то параметр можно назвать `МассивСчетовНаОплату` или `СчетаНаОплату`. Если `МножественныйВыбор` отключен, то параметр можно назвать `СчетНаОплату` или `ССылкаСчетаНаОплату`.
		2-й параметр обработчика команды `ПараметрыВыполнения` имеет тип `Структура` со следующими полями:
		• `ОписаниеКоманды` – `Структура` – описание команды (ключи соответствуют колонкам этой таблицы):
		• `Идентификатор` – `Строка` – идентификатор команды.
		• `Представление` – `Строка` – представление команды в форме.
		• `ДополнительныеПараметры` – `Неопределено`, `ФиксированнаяСтруктура` – дополнительные параметры команды.
		• `Форма` – `ФормаКлиентскогоПриложения` – форма, из которой вызвана команда.
		• `ЭтоФормаОбъекта` – `Булево` – `Истина`, если команда вызвана из формы объекта.
		• `Источник` – `ТаблицаФормы`, `ДанныеФормыСтруктура` – объект или список формы с полем `Ссылка`.

`ИмяФормы` (обязательный, если не указан `Обработчик`)	`Строка`	Имя формы, которую требуется получить для выполнения команды. Если `Обработчик` не указан, то у формы вызывается метод `Открыть`.
`ПараметрыФормы` (необязательный)	`Структура`, `Неопределено`.	Параметры формы, указанной в `ИмяФормы`.
`ДополнительныеПараметры` (необязательный)	`Структура`, `Неопределено`.	Дополнительные параметры, которые могут быть считаны в обработчике команды.

Параметры команд Заполнения объектов, настройки видимости и доступности

Параметр	Тип	Описание
`ТипПараметра` (необязательный)	`ОписаниеТипов`	Типы объектов, для которых предназначена эта команда. Используется, когда обработка (или другой поставщик команд) подключена к нескольким объектам конфигурации, для уточнения списка объектов, к которым подключена конкретная команда.
`ВидимостьВФормах` (необязательный)	`Строка`	Имена форм, в которых должна отображаться команда, через запятую. Используется для уточнения состава форм, к которым подключена конкретная команда. Если параметр не указан, то команда будет отображаться во всех формах. Пример: `Команда.ВидимостьВФормах = "ФормаДокумента"`
`ФункциональныеОпции` (необязательный)	`Строка`	Имена функциональных опций, определяющих видимость команды, через запятую.
`УсловияВидимости` (необязательный)	`Массив`	Определяет видимость команды в зависимости от контекста. Для регистрации условий следует использовать процедуру `ПодключаемыеКоманды.ДобавитьУсловиеВидимостиКоманды`. Условия объединяются по «И».

Параметры команд Заполнения объектов, настройки процесса выполнения

Параметр	Тип	Описание
`МножественныйВыбор` (необязательный)	`Булево`	Если `Истина`, то команда поддерживает множественный выбор и в 1 параметре обработчика команды будет передан массив ссылок. Если `Ложь`, то команда поддерживает только одиночный выбор и в 1 параметре обработчика команды будет передана 1 ссылка.
`РежимЗаписи` (необязательный)	`Строка`	Настройки дополнительных проверок и действий, связанных с записью объекта, выполняемых перед обработчиком команды: • `НеЗаписывать` – объект не записывается, а в параметрах обработчика вместо ссылок передается вся форма. В этом режиме рекомендуется работать напрямую с формой, которая передается в структуре 2-го параметра обработчика команды. • `ЗаписыватьТолькоНовые` – записывать только новые объекты. • `Записывать` – записывать новые и модифицированные объекты. • `Проводить` – проводить документы. Например: `Команда.РежимЗаписи = "НеЗаписывать";` Перед записью и проведением у пользователя запрашивается подтверждение. Значение по умолчанию: `Записывать.`
`ТребуетсяРаботаСФайлами` (необязательный)	`Булево`	Если `Истина`, то в веб-клиенте перед выполнением команды предлагается установить расширение для работы с 1С:Предприятием.

Примеры команд и их обработчиков см. в обработке `_ДемоПодключаемыеКомандыЗаполнениеКонтрагентов` расширения `_ДемоПодключаемыеКоманды` демонстрационной конфигурации.

Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд создания на основании по шаблону:

- Имя: `ПодменюЗаполнить`.
- Заголовок: **Заполнить**.
- Вид: `Подменю`.
- Отображение: `Картинка`.
- Картинка: `ЗаполнитьФорму` (картинка из конфигурации).

Помимо основного подменю в форме могут быть размещены дополнительные подменю (например, для заполнения табличных частей). Имена этих подменю указываются в свойстве `Группа` тех команд, которые должны выводиться в этих подменю. Имена подменю заполнения табличных частей рекомендуется образовывать путем добавления префикса с именем табличной части, например, `ТоварыПодменюЗаполнить`.

Расширение видов подключаемых команд

В стандартной поставке предусмотрены следующие виды подключаемых команд: создание на основании, печатные формы, отчеты и команды заполнения. Однако есть возможность добавлять и собственные виды подключаемых команд.

Для этого в модуле `ПодключаемыеКомандыПереопределяемый` необходимо:

- В процедуре `ПриОпределенииВидовПодключаемыхКоманд` определить собственные виды подключаемых команд и задать умолчания для вывода подменю с командами этих видов. Например:

```
Процедура ПриОпределенииВидовПодключаемыхКоманд(ВидыПодключаемыхКоманд) Экспорт

    Вид = ВидыПодключаемыхКоманд.Добавить();
    Вид.Имя          = "Мотиваторы";
    Вид.ИмяПодменю   = "ПодменюМотиваторов";
    Вид.Заголовок    = НСтр("ru = 'Мотиваторы'");

КонецПроцедуры
```

- В процедуре `ПриОпределенииСоставНастроекПодключаемыхОбъектов` описать ключ (например, типа `Булево`), который будет служить маркером того, что объект поставляет команды данного вида. Например:

```
Процедура ПриОпределенииСоставНастроекПодключаемыхОбъектов(НастройкиПрограммногоИнтерфейса) Экспорт

    Настройка = НастройкиПрограммногоИнтерфейса.Добавить();
    Настройка.Ключ           = "ДобавитьМотиваторы";
    Настройка.ОписаниеТипов = Новый ОписаниеТипов("Булево");
    Настройка.ВидыПодключаемыхОбъектов = "Обработка";

КонецПроцедуры
```

- В процедуре `ПриОпределенииКомандПодключенныхКОбъекту` добавить команды данного вида, которые соответствуют переданному контексту. Например:

```
Процедура ПриОпределенииКомандПодключенныхКОбъекту(НастройкиФормы, Источники, ПодключенныеОтчетыИОбработки, Команды) Экспорт

    <МодульПодсистемы>.ПриОпределенииКомандПодключенныхКОбъекту(НастройкиФормы, Источники, ПодключенныеОтчетыИОбработки, Команды);

КонецПроцедуры
```

- Описать в документации, как разрабатывать команды данного вида.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Поиск и удаление дублей

Подсистема **Поиск и удаление дублей** позволяет искать и удалять дубли элементов (справочников и т. п.), заменяя во всех местах использования дублирующие элементы выбранным.

Настройка

В случае если в конфигурации не используется подсистема **Настройки программы**, то в рабочем месте администратора приложения необходимо разместить обработку `ПоискИУдалениеДублей`. См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

Объединение дублей и поиск мест использования элементов списков

Нужно принять решение по поводу состава объектов конфигурации, в списках которых востребована возможность объединения дублей и мест использования выбранных элементов. Это могут быть справочники, планы видов характеристик, планы счетов и планы видов расчета, в которых вследствие ручного ввода или синхронизации данных с другими приложениями часто возникают дубли и необходимо их оперативное устранение.

При использовании совместно с подсистемой Подключаемые команды имеется возможность вывести команды **Объединить выделенные...** и **Заменить выделенные...** в формах списков и журналов в подменю **Ещё - Сервис**. Для этого необходимо перечислить объекты в процедуре `ПриОпределенииОбъектовСКомандамиОбъединенияДублейЗаменыСсылок` модуля `ПоискИУдалениеДублейПереопределяемый`:

```
Процедура ПриОпределенииОбъектовСКомандамиОбъединенияДублейЗаменыСсылок(Объекты) Экспорт
    // _Демо начало примера
    Объекты.Добавить(Метаданные.Справочники._ДемоНоменклатура);
    // _Демо конец примера
КонецПроцедуры
```

Если подсистема Подключаемые команды не используется, то есть возможность добавить команды `ОбъединитьВыделенные` (синоним **Объединить выделенные...**) и `ПоказатьМестаИспользования` (синоним **Места использования**) непосредственно в формы:

```
// СтандартныеПодсистемы.ПоискИУдалениеДублей
&НаКлиенте
Процедура ОбъединитьВыделенные(Команда)

    ПоискИУдалениеДублейКлиент.ОбъединитьВыделенные(Элементы.Список);

КонецПроцедуры
&НаКлиенте
Процедура ПоказатьМестаИспользования(Команда)

    ПоискИУдалениеДублейКлиент.ПоказатьМестаИспользования(Элементы.Список);

КонецПроцедуры
// Конец СтандартныеПодсистемы.ПоискИУдалениеДублей
```

Рекомендуется размещать указанные команды в подменю **Еще** и в контекстном меню списка. Пример внедрения см. в справочнике `_ДемоФизическиеЛица`.

При использовании совместно с подсистемами Варианты отчетов и Подключаемые команды имеется возможность автоматически вывести команду `ПоказатьМестаИспользования` (вместо ручного добавления) во всех формах, в которых встроено подменю Отчеты. Для этого в модуле `ВариантыОтчетовПереопределяемый` в процедуре `ПередДобавлениемКомандОтчетов` вставить вызов по шаблону:

```
Процедура ПередДобавлениемКомандОтчетов(КомандыОтчетов, Параметры, СтандартнаяОбработка) Экспорт
    Отчеты.МестаИспользованияСсылок.ДобавитьКомандуМестаИспользования(КомандыОтчетов);
КонецПроцедуры
```

Ограничения замены ссылок, поиска и удаления дублей

При использовании обработки `ЗаменаСсылок` и процедуры `ЗаменитьСсылки` общего модуля `ОбщегоНазначения` существует возможность контроля допустимости замены одной ссылки на другую. Для этого необходимо добавить в модуль менеджера контролируемого объекта (справочника, плана видов характеристик и т. п.) функцию `ВозможностьЗаменыЭлементов`. В ней реализовать алгоритм, который с прикладной точки зрения анализирует предлагаемые замены и возвращает описание ошибки о недопустимости замены одной ссылки на другую.

Также предусмотрена возможность реализовать собственную логику поиска дублей при вызове функции `НайтиДублиЭлемента` общего модуля `ПоискИУдалениеДублей` и при использовании обработки `ПоискИУдалениеДублей`. Для этого необходимо определить в модуле менеджера интересующего нас объекта две процедуры:

- процедуру `ПараметрыПоискаДублей`, в которой можно переопределить параметры поиска дублей, заданные пользователем или при программном поиске;
- процедуру `ПриПоискеДублей`, в которой реализовать собственную логику определения дублей.

Пример реализации, описания и использования этих функций можно найти в модуле менеджера справочника `_ДемоНоменклатура`. Пример использования программного интерфейса – в обработчике записи модуля формы этого же справочника.

Замена связанных подчиненных объектов

При использовании обработки `ЗаменаСсылок` и процедуры `ЗаменитьСсылки` общего модуля `ОбщегоНазначения` может возникнуть необходимость в местах использования менять не только заменяемые ссылки, но и связанные объекты. Например, вместе с контрагентом заменить банковский счет, а при замене номенклатуры свернуть связанные с ней ключи аналитики.

Для этого в процедуре `ПриОпределенииСвязейПодчиненныхОбъектов` модуля `ОбщегоНазначенияПереопределяемый` нужно описать перечень подчиненных объектов и их связи:

```
Процедура ПриОпределенииПодчиненныхОбъектов (ПодчиненныеОбъекты) Экспорт

    ПодчиненныйОбъект = ПодчиненныеОбъекты.Добавить();
    ПодчиненныйОбъект.ПодчиненныйОбъект = Метаданные.Справочники.БанковскиеСчета;
    ПодчиненныйОбъект.ПоляСвязей = "Владелец, НомерСчета, БИКБанка";
    ПодчиненныйОбъект.ПриПоискеЗаменыСсылок = "Справочники.БанковскиеСчета";

    ПодчиненныйОбъект = ПодчиненныеОбъекты.Добавить();
    ПодчиненныйОбъект.ПодчиненныйОбъект = Метаданные.Справочники.КлючиАналитикиНоменклатуры;
    ПодчиненныйОбъект.ПоляСвязей = Справочники.КлючиАналитикиНоменклатуры.КлючевыеРеквизиты();
    ПодчиненныйОбъект.ПриПоискеЗаменыСсылок = "Справочники.КлючиАналитикиНоменклатуры";
    ПодчиненныйОбъект.ВыполнятьАвтоматическийПоискЗаменСсылок = Истина;

КонецПроцедуры
```

Параметры связи подчиненных объектов

Параметр	Тип	Описание
`ПодчиненныйОбъект` (обязательный)	`ОбъектМетаданных`	Метаданные подчиненного объекта.
`ПоляСвязей` (обязательный)	`Строка`, `Структура`, `Соответствие`.	Список реквизитов подчиненного объекта, в которых содержаться ссылки на основные объекты.

`ПриПоискеЗаменыСсылок` (необязательный)	`Строка`	Процедура вызывается в случае, если автоматический поиск отключен или автоматически не удалось подобрать значение для замены.
		Если параметр заполнен, ожидается процедура с именем `ПриПоискеЗаменыСсылок` по шаблону:
		`Процедура ПриПоискеЗаменыСсылок(ПарыЗамен,НеобработанныеЗначенияОригиналов) Экспорт`
		`// Код подбора замен`
		`КонецПроцедуры`
		Параметры:
		• `ПарыЗамен` - Соответствие - содержит полный перечень замен;
		• `НеобработанныеЗначенияОригиналов` - Массив - сведений о подчиненном объекте, для которого не удалось подобрать замену.
		См. Пример в модуле менеджера справочника `_ДемоБанковскиеСчета`.

`ВыполнятьАвтоматическийПоискЗаменСсылок` (необязательный)	`Булево`	Если `Истина`, то будет выполнена попытка найти замену для подчиненного объекта по совпадению полей связи с учетом замен основных объектов.

См. пример в демонстрационной конфигурации.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы нужно использовать следующие роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП`, `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**) Просмотр отчета **Места использования**, замена и объединение элементов, разрешенных правами доступа.
2.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Удаление дублей элементов с заменой во всех местах использования дублирующих элементов выбранным.

Использование при разработке конфигурации

Контроль замены ссылок при записи объектов и выполнение связанных действий

При замене ссылок в документах документы записываются без перепроведения, чтобы не нарушать последовательности. Как следствие, не срабатывает событие `ОбработкаПроведения` и вся прикладная логика проведения по регистрам. Подробнее см. раздел Контроль замены ссылок при записи объектов и выполнение связанных действий.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Полнотекстовый поиск

Подсистема **Полнотекстовый поиск** предназначена для включения в конфигурации возможности полнотекстового поиска данных. Индексация данных для полнотекстового поиска может проводиться регламентными заданиями `СлияниеИндексаППД` и `ОбновлениеИндексаППД`.

Настройка

Для использования подсистемы в конфигурации необходимо назначить общую форму `ФормаПоиска` основной формой поиска конфигурации. Кроме того, на начальной странице рекомендуется разместить форму `УпрощеннаяФорма` обработки `ПолнотекстовыйПоискВДанных`.

Размещение настроек подсистемы в собственных формах

Если в конфигурации не используется подсистема **Настройки программы**, то для открытия настроек полнотекстового поиска в интерфейсе администратора приложения необходимо предусмотреть команду, вызывающую процедуру `ПоказатьУправлениеПолнотекстовымПоискомИИзвлечениемТекстов` общего модуля `ПолнотекстовыйПоискКлиент`.

При совместном внедрении с подсистемой **Работа с файлами** форма `УправлениеПолнотекстовымПоискомИИзвлечениемТекстов` обработки `ПолнотекстовыйПоискВДанных` также позволяет настраивать автоматическое извлечение текстов из файлов для их включения в индекс полнотекстового поиска.

Для размещения настроек подсистемы в произвольной форме следует:

- создать реквизит формы `ИспользоватьПолнотекстовыйПоиск` типа `Число` для управления флажком настроек, который следует разместить в элементах формы;
- в обработчике события формы `ПриСозданииНаСервере` вызвать процедуру `ЗначениеФлажкаИспользоватьПоиск` модуля `ПолнотекстовыйПоискСервер`.
- в обработчике события формы `ОбработкаОповещения` вызвать процедуру `ОбработкаОповещенияИзмененияФлажкаИспользоватьПоиск` модуля `ПолнотекстовыйПоискКлиент`.
- при изменении реквизита формы `ИспользоватьПолнотекстовыйПоиск` вызвать процедуру `ПриИзмененииФлажкаИспользоватьПоиск` модуля `ПолнотекстовыйПоискКлиент`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Полнотекстовый поиск** следует использовать роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Использование возможностей полнотекстового поиска. Обновление и слияние индекса ППД.
2.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение полнотекстового поиска.

Использование при разработке конфигурации

Настройка обмена данными

Константу `ИспользоватьПолнотекстовыйПоиск` рекомендуется включать только в состав начального образа подчиненного узла распределенной информационной базы (РИБ) и автономного рабочего места (см. раздел Особенности создания начального образа подчиненного узла распределенной ИБ).

Получение файлов из Интернета

Подсистема **Получение файлов из Интернета** добавляет в конфигурацию программный интерфейс для получения файлов из сети Интернет по протоколам HTTP, HTTPS и FTP и сохранения полученных файлов на клиенте, сервере или во временном хранилище.

Настройка

Для использования подсистемы в конфигурации необходимо:

1. В случае если в конфигурации не используется подсистема **Настройки программы**, в командном интерфейсе администратора разместить команду для открытия общей формы `ПараметрыПроксиСервера` по настройке прокси-сервера для доступа к Интернету на сервере **1С:Предприятия**. Код команды:

   ```
   ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера");
   ```

2. Разместить в форме персональных настроек работы с информационной базой (для пользователя информационной базы) команду настройки прокси-сервера для доступа к Интернету с клиентского места. Код команды:

   ```
   ОткрытьФорму("ОбщаяФорма.ПараметрыПроксиСервера", Новый Структура("НастройкаПроксиНаКлиенте", Истина));
   ```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Получение файлов из Интернета** следует использовать роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Получение файлов из Интернета, настройка параметров прокси-сервера на стороне клиентского приложения.
2.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Настройка параметров прокси-сервера для сервера **1С:Предприятия**.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор • Настройка серверного прокси-сервера • Получение файлов из Интернета	• `АдминистраторСистемы` (из подсистемы **Базовая функциональность**), • `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Пользователь с ограниченными правами • Настройка клиентского прокси-сервера • Получение файлов из Интернета	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**)

Использование при разработке конфигурации

При создании объектов `HTTPСоединение`, `FTPСоединение`, `WSОпределения` и `WSПрокси` следует указывать настройки прокси-сервера, полученные функцией `ПолучениеФайловИзИнтернета.ПолучитьПрокси`.

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы не должны включаться в планы обмена, предназначенные для синхронизации данных между различными приложениями, для организации распределенной информационной базы (РИБ) и автономного рабочего места.

Пользователи

Подсистема **Пользователи** предназначена для просмотра и редактирования списка пользователей и внешних пользователей системы (элементы справочников **Пользователи** и **Внешние пользователи**), который синхронизируется со списком пользователей информационной базы.

Внешние пользователи предназначены для организации доступа к информационной системе тем пользователям, которые представлены в ней одним из объектов (например, **Сотрудники**, **Партнеры**, **Физические лица** и др.). Подсистема предоставляет средства для сопоставления внешних пользователей с объектами информационной базы.

Настройка

Принять решение по поводу состава объектов конфигурации, которые должны быть сопоставлены внешним пользователям системы. Например, справочники **Физические лица**, **Сотрудники**, **Партнеры**. Если такие объекты есть, то перечислить эти допустимые типы внешних пользователей:

- в определяемых типах `ВнешнийПользователь` (ссылки) и `ВнешнийПользовательОбъект` (объекты);
- в свойстве `Тип` параметра команды команды `ВнешнийДоступ` справочника `ВнешниеПользователи`;
- в определяемом типе `Пользователь` (ссылки). В состав этого типа также входит тип `СправочникСсылка.Пользователи`.

В справочнике `Пользователи` предусмотрены реквизиты `ФизическоеЛицо` и `Подразделение`, которые могут быть дополнительно задействованы в конфигурации. Для того чтобы дать возможность вводить значения этих реквизитов, необходимо задать состав определяемых типов `ФизическоеЛицо` и `Подразделение`. Если же ввод этих реквизитов не требуется, то состав указанных определяемых типов следует оставить пустым (и тогда в форме элемента справочника `Пользователи` реквизиты будут отсутствовать).

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, в командном интерфейсе администратора приложения необходимо разместить:

- Справочник `Пользователи`:
  - для просмотра списка, свойств и выбора пользователей (обычными пользователями);
  - для просмотра и редактирования состава и свойств пользователей (ответственными за список пользователей и администраторами).
- Справочник `ВнешниеПользователи`:
  - для просмотра списка, свойств и выбора внешних пользователей (обычными пользователями);
  - для просмотра и редактирования состава и свойств внешних пользователей (ответственными за список внешних пользователей и администраторами).
- `ИспользоватьВнешнихПользователей` – для включения/выключения возможности использования внешних пользователей (администраторами).
- `ИспользоватьГруппыПользователей` – для включения/выключения возможности использования групп пользователей и групп внешних пользователей (администраторами).

См. пример в форме `НастройкиПользователейИПрав` обработки `ПанельАдминистрированияБСП`.

Для доступа пользователей к своим персональным настройкам необходимо создать и разместить в форме персональных настроек команду **Сведения о пользователе**. См. пример размещения в демонстрационной конфигурации, в форме `_ДемоМоиНастройки` на закладке `Общие`.

Для удобства работы внешних пользователей на рабочий стол рекомендуется вынести формы тех подсистем, которые будут доступны внешнему пользователю после успешной аутентификации. Например, это может быть форма доступных анкет респондента или форма заказов покупателя.

Назначение ролей

Роли отбираются по назначению при выборе в режиме **1С:Предприятие**. Например, роли только для внешних пользователей недоступны для пользователей. При входе пользователя в **1С:Предприятие** выполняется проверка ролей в целях безопасности (в модели сервиса проверяются роли пользователя, в локальном режиме проверяются роли внешнего пользователя). Также назначения ролей проверяются в профилях групп доступа (см. подсистему Управление доступом).

При разработке ролей необходимо указывать роли специального назначения в процедуре `ПриОпределенииНазначенияРолей` общего модуля `ПользователиПереопределяемый`. Все остальные роли не требуется указывать. Назначение ролей подробно описано в комментарии к процедуре.

```
// Позволяет указать роли специального назначения. Все остальные роли не требуется указывать -
// это роли, которые предназначены для любых пользователей, кроме внешних пользователей.
//
// Параметры:
//  НазначениеРолей - Структура - со свойствами:
//   * ТолькоДляАдминистраторовСистемы - Массив - имена ролей, которые при выключенном разделении
//     предназначены для любых пользователей, кроме внешних пользователей, а в разделенном режиме
//     предназначены только для администраторов сервиса, например:
//       Администрирование, ОбновлениеКонфигурацииБазыДанных, АдминистраторСистемы,
//     а также все роли с правами:
//       Администрирование,
//       Администрирование расширений конфигурации,
//       Обновление конфигурации базы данных.
//     Такие роли, как правило, существуют только в БСП и не встречаются в прикладных решениях.
//
//   * ТолькоДляПользователейСистемы - Массив - имена ролей, которые при выключенном разделении
//     предназначены для любых пользователей, кроме внешних пользователей, а в разделенном режиме
//     предназначены только для неразделенных пользователей (сотрудников технической поддержки сервиса и
//     администраторов сервиса), например:
//       ДобавлениеИзменениеАдресныхСведений, ДобавлениеИзменениеБанков,
//     а также все роли с правами изменения неразделенных данных и следующими правами:
//       Толстый клиент,
//       Внешнее соединение,
//       Automation,
//       Режим все функции,
//       Интерактивное открытие внешних обработок,
//       Интерактивное открытие внешних отчетов.
//     Такие роли в большей части существует в БСП, но могут встречаться и в прикладных решениях.
//
//   * ТолькоДляВнешнихПользователей - Массив - имена ролей, которые предназначены
//     только для внешних пользователей (роли со специально разработанным набором прав), например:
//       ДобавлениеИзменениеОтветовНаВопросыАнкет, БазовыеПраваВнешнихПользователейБСП.
//     Такие роли существуют и в БСП, и в прикладных решениях (если используются внешние пользователи).
//
//   * СовместноДляПользователейИВнешнихПользователей - Массив - имена ролей, которые предназначены
//     для любых пользователей (и внутренних, и внешних, и неразделенных), например:
//       ЧтениеОтветовНаВопросыАнкет, ИспользованиеВариантовОтчетов.
//     Такие роли существуют и в БСП, и в прикладных решениях (если используются внешние пользователи).
//
Процедура ПриОпределенииНазначенияРолей(НазначениеРолей) Экспорт

    // ТолькоДляВнешнихПользователей.
    НазначениеРолей.ТолькоДляВнешнихПользователей.Добавить(
        Метаданные.Роли._ДемоОплатаСчетовВнешнимиПользователями.Имя);

    НазначениеРолей.ТолькоДляВнешнихПользователей.Добавить(
         Метаданные.Роли._ДемоЧтениеДанныхОбъектовАвторизации.Имя);

    // СовместноДляПользователейИВнешнихПользователей.
    НазначениеРолей.СовместноДляПользователейИВнешнихПользователей.Добавить(
        Метаданные.Роли._ДемоЧтениеДанныхДляОтветовНаВопросыАнкет.Имя);

КонецПроцедуры
```

> **Внимание!**
>
> Для того чтобы изменения в реализации этой процедуры вступили в силу сразу при разработке и отладке конфигурации, необходимо обновить вспомогательные данные, подробнее см. раздел Обновление вспомогательных данных во время разработки.

Отображение внешних пользователей в списках

В конфигурациях, в которых партнерам или сотрудникам предоставляется доступ к приложению извне, рекомендуется в форме списка справочника добавить колонку, показывающую наличие внешнего доступа у этого партнера или сотрудника. Для этого, например, в форме списка справочника `_ДемоПартнеры`, являющегося объектом авторизации справочника `ВнешниеПользователи`, необходимо в запросе динамического списка сделать левое соединение со справочником `ВнешниеПользователи` (по реквизиту `ОбъектАвторизации`).

Было:

```
ВЫБРАТЬ РАЗРЕШЕННЫЕ
    Справочник_ДемоПартнеры.Ссылка КАК Ссылка,
    Справочник_ДемоПартнеры.Наименование КАК Наименование
ИЗ
    Справочник._ДемоПартнеры КАК Справочник_ДемоПартнеры
```

Стало:

```
ВЫБРАТЬ РАЗРЕШЕННЫЕ
    -1 КАК ВнешнийДоступНомерКартинки,
    НЕ ВнешниеПользователи.Ссылка ЕСТЬ NULL
        И НЕ ВнешниеПользователи.Недействителен
        И НЕ ВнешниеПользователи.ПометкаУдаления КАК ВнешнийДоступ,
    Справочник_ДемоПартнеры.Ссылка КАК Ссылка,
    Справочник_ДемоПартнеры.Наименование КАК Наименование
ИЗ
    Справочник._ДемоПартнеры КАК Справочник_ДемоПартнеры
        {ЛЕВОЕ СОЕДИНЕНИЕ Справочник.ВнешниеПользователи КАК ВнешниеПользователи
        ПО Справочник_ДемоПартнеры.Ссылка = ВнешниеПользователи.ОбъектАвторизации}
```

В событие `ПриСозданииНаСервере` разместить следующий код:

```
ВнешниеПользователи.НастроитьОтображениеСпискаВнешнихПользователей(ЭтотОбъект);
```

В событие элемента таблицы динамического списка `ПриПолученииДанныхНаСервере` разместить следующий код:

```
ВнешниеПользователи.СписокВнешнихПользователейПриПолученииДанныхНаСервере(ИмяЭлемента, Настройки, Строки);
```

Под элементом таблицы динамического списка разместить группу `ВнешнийДоступЛегенда` с описанием состояний.

См. пример в формах `ФормаСписка` и `ФормаВыбора` справочников `_ДемоПартнеры` и `_ДемоКонтактныеЛицаПартнеров` в демонстрационной базе.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Пользователи» без подсистемы «Управление доступом».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Пользователи** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**). Чтение списка пользователей без ограничений.
2.	`ДобавлениеИзменениеПользователей` (используется совместно с ролью `БазовыеПраваБСП`). Добавление и изменение пользователей и групп пользователей без настройки доступа к приложению. Дополнительная роль, предназначенная для секретарей, менеджеров по персоналу: добавление новых пользователей при приеме на работу, исправление ФИО, контактной информации и других нормативных свойств пользователя, создание новых групп пользователей. Сокращает рутинную нагрузку на администратора в крупных компаниях.
3.	`ЧтениеВнешнихПользователей` (используется совместно со вспомогательной ролью `ЧтениеДанныхОбъектовАвторизации`). Чтение внешних пользователей и групп внешних пользователей (просмотр списка внешних пользователей без ограничений).
4.	`ДобавлениеИзменениеВнешнихПользователей` (используется совместно со вспомогательной ролью `ЧтениеДанныхОбъектовАвторизации`). Добавление и изменение внешних пользователей и групп внешних пользователей без настройки доступа к приложению (просмотр списка внешних пользователей без ограничений). Дополнительная роль, предназначенная для помощников руководителей менеджеров по продажам и закупкам: добавление новых внешних пользователей при новых контактах, создание новых групп внешних пользователей. Сокращает рутинную нагрузку на администратора в крупных компаниях.
5.	`ПолныеПрава` (из подсистемы **Базовая функциональность**). Добавление и изменение пользователей, групп пользователей, внешних пользователей, групп внешних пользователей, настроек входа и прав доступа. Изменение пароля, имени для входа и остальных свойств любого пользователя и внешнего пользователя. Включение и отключение использования внешних пользователей. Удаление объектов подсистемы, помеченных на удаление.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным, которые не относятся к подсистеме **Пользователи**, но требуются для работы с ней.

№	Вспомогательные роли и их назначение
1.	`<ЧтениеДанныхОбъектовАвторизации>` Просмотр сведений об объектах авторизации.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	**Администратор**	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	**Ответственный за список пользователей (дополнительно)**	• `ДобавлениеИзменениеПользователей`, • `Подсистема_ДемоАдминистрирование` (просмотр подсистемы **Администрирование** подсистемы **Настройки программы**)
3.	**Ответственный за список внешних пользователей (дополнительно)**	• `ДобавлениеИзменениеВнешнихПользователей`, • `ЧтениеДанныхОбъектовАвторизации`, • `Подсистема_ДемоАдминистрирование` (просмотр подсистемы **Администрирование** подсистемы **Настройки программы**)
4.	**Пользователь, работающий с внешними пользователями (дополнительно)**	• `ЧтениеВнешнихПользователей`.
5.	**Внешний пользователь**	• `БазовыеПраваВнешнихПользователейБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**)

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Пользователи» без подсистемы «Контактная информация».
- Внедрение подсистемы «Пользователи» без подсистемы «Свойства».

Использование при разработке конфигурации

При создании новых прикладных объектов метаданных нужно повторно выполнять соответствующую часть процедуры настройки, описанной выше (например, пересматривать состав типов объектов авторизации).

Для получения ссылки на пользователя, который произвел вход в информационную базу, рекомендуется использовать функцию `АвторизованныйПользователь` общего модуля `ПользователиКлиентСервер`, которая возвращает ссылку на элемент справочника `Пользователи` или элемент справочника `ВнешниеПользователи`.

В коде, вызов которого предполагается только в сеансе пользователей, рекомендуется использовать функцию `ТекущийПользователь` общего модуля `ПользователиКлиентСервер`, которая возвращает ссылку на элемент справочника `Пользователи`. При вызове функции в сеансе внешнего пользователя будет вызвано исключение.

В коде, вызов которого предполагается только в сеансе внешних пользователей, рекомендуется использовать функцию `ТекущийВнешнийПользователь`, которая возвращает ссылку на элемент справочника `ВнешниеПользователи`. При вызове функции в сеансе пользователя будет вызвано исключение.

Недействительные пользователи

В формах пользователя и внешнего пользователя предусмотрен флажок `Недействителен`, связанный с реквизитом `Недействителен` (по умолчанию – `Ложь`). При установке флажка `Недействителен` пользователь скрывается в списке пользователей, списке выбора пользователей, а также при подборе по строке. Для того чтобы увидеть всех пользователей в списке пользователей и списке выбора пользователей, надо установить флажок **Показывать недействительных пользователей**.

При обычном использовании справочников описанное поведение поддерживается автоматически. При открытии списка пользователей или списка выбора пользователей флажок **Показывать недействительных пользователей** снят. При подборе по строке в параметры получения данных выбора автоматически вставляется отбор `Недействителен` = `Ложь` (если отбор для реквизита `Недействителен` еще не указан).

При особых случаях использования следует поддержать описанное поведение самостоятельно:

- При разработке своего списка выбора пользователей. Например, форма, аналогичная общей форме `ВыборИсполнителяБизнесПроцесса`.
- При подготовке отчетов, в которые могут попасть недействительные пользователи. Например, в конфигурации `Документооборот` в отчете по задачам определяется, к какому отделу относится задача, выданная роли. Отдел определяется по пользователям, входящим в роль. Недействительные пользователи будут исключены из этого списка – их отделы не попадут в отчет.
- При обращении к справочнику на языке запросов или из кода.

Служебные пользователи

В справочнике `Пользователи` предусмотрен реквизит `Служебный`, который программно устанавливается для скрытия таких пользователей из интерфейса приложения в специальных случаях:

- Для пользователей, от имени которых подключаются к приложению http-сервисы, веб-сервисы, внешние соединения и др.
- Для пользователей со специальными правами, от имени которых запускаются регламентные задания, когда не подходит пользователь по умолчанию. Например, требуется обращение к системе взаимодействия или выполнение с ограниченными правами.
- В модели сервиса для скрытия администраторов сервиса в областях данных.

Если значение реквизита `Служебный` установлено в `Истина`, пользователь скрывается в списке пользователей, списке выбора пользователей, а также при подборе по строке. В других пользовательских интерфейсах это поведение следует реализовать самостоятельно.

Служебные пользователи создаются только программно, после чего признак `Служебный` изменять недопустимо. Для них также программно необходимо назначать и поддерживать в актуальном состоянии роли (при переходе на новую версию и в других сценариях). При использовании подсистемы **Управление доступом** состав ролей служебных пользователей остается неизменным в отличие от других пользователей, роли которых рассчитываются и назначаются автоматически при включении или исключении из групп доступа.

При создании служебных пользователей для регламентных заданий рекомендуется отключать аутентификацию таких пользователей, а также вместо назначения ролей администратора "точечно" включать привилегированный режим для выполнения требуемых действий.

Внешние пользователи

При 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 `ПользователиКлиентСервер`:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    Если Не ПользователиКлиентСервер.ЭтоСеансВнешнегоПользователя() Тогда
        Отказ = Истина;
        Возврат;
    КонецЕсли;
    …
КонецПроцедуры
```

Настройка регистрации события журнала Доступ.Доступ

С помощью события журнала регистрации **Доступ.Доступ** можно реализовать аудит доступа к интересующим данным. При этом не следует записывать настройки этого события напрямую с помощью метода платформы `УстановитьИспользованиеСобытияЖурналаРегистрации`, а вместо этого рекомендуется использовать программный интерфейс:

- процедура `ПриОпределенииНастроекРегистрацииСобытийДоступаКДанным` в общем модуле `ПользователиПереопределяемый`,
- функция `НастройкиРегистрацииСобытийДоступаКДанным` общего модуля `Пользователи`,
- процедура `ОбновитьНастройкиРегистрацииСобытийДоступаКДанным` в общем модуле `Пользователи`.

Это позволяет каждому механизму-потребителю хранить свои настройки отдельно, а при их изменении применять к информационной базе общий результат без случайного перетирания настроек других механизмов. Для этого в процедуру `ПользователиПереопределяемый.ПриОпределенииНастроекРегистрацииСобытийДоступаКДанным` разместить вызов своей процедуры по добавлению своей части настроек, а для их установки вызывать поцедуру `Пользователи.ОбновитьНастройкиРегистрацииСобытийДоступаКДанным` (которая централизованно собирает все настройки и устанавливает их).

При ошибке установки настроек `Пользователи.ОбновитьНастройкиРегистрацииСобытийДоступаКДанным` автоматически удаляет из настроек несуществующие имена полей, повторяет попытку установки и записывает в журнал регистрации событие **Пользователи.Ошибка настройки события Доступ.Доступ** с подробной информации и не вызывается исключение.

Не следует самостоятельно отключать регистрацию события **Доступ.Доступ**, достаточно отключить (не добавлять) свою часть настроек в `ПользователиПереопределяемый.ПриОпределенииНастроекРегистрацииСобытийДоступаКДанным` и вызвать `Пользователи.ОбновитьНастройкиРегистрацииСобытийДоступаКДанным`.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме следующих:

- константа `НастройкиВходаПользователей`,
- константа `РегистрироватьИзмененияПравДоступа`,
- регистр сведений `СведенияОПользователях`.

В планах обмена распределенной информационной базы (РИБ) рекомендуется отключать регистрацию изменений для следующих объектов метаданных подсистемы (см. также раздел Особенности создания начального образа подчиненного узла распределенной ИБ):

- регистр сведений `ИерархияГруппПользователей`,
- регистр сведений `СоставыГруппПользователей`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется также исключать следующие объекты метаданных:

- константа `НастройкиВходаПользователей`,
- константа `РегистрироватьИзмененияПравДоступа`,
- регистр сведений `ИерархияГруппПользователей`,
- регистр сведений `СведенияОПользователях`,
- регистр сведений `СоставыГруппПользователей`.

Префиксация объектов

Подсистема **Префиксация объектов** предназначена для автоматического назначения префиксов объектам с учетом настроек приложения. Префиксация объектов ведется в разрезах информационных баз и элементов справочника `Организации`.

Настройка

Настройка подсистемы **Префиксация объектов** различается в зависимости от внедрения совместно с подсистемой **Обмен данными** и наличия в конфигурации справочника `Организации`. Настройка подсистемы сводится к созданию подписок на события для задания номера (или кода) объекта и подписки для очистки номера объекта.

Создание подписок для установки префикса в номере (коде) объекта

Следует создать необходимое количество подписок на события для задания номера или кода объекта при его записи, если номер или код не заполнены. Значения свойств подписок задать согласно таблице.

Имя свойства	Описание
`Имя`	Имя следует задавать для удобства визуального восприятия подписки и с учетом ограничений, налагаемых на имена объектов метаданных, например `УстановитьПрефиксИнформационнойБазыИОрганизацииНомеруДокумента`.
`Источник`	Источниками событий подписки могут быть только объекты типов: `Документы`, `Справочники`, `ПланыВидовХарактеристик`, `БизнесПроцессы`. В качестве источников для одной подписки могут быть использованы объекты только одного типа.
`Событие`	`ПриУстановкеНовогоКода` или `ПриУстановкеНовогоНомера`
`Обработчик`	В качестве процедур – обработчиков подписок следует выбирать экспортные процедуры общего модуля `ПрефиксацияОбъектовСобытия`.

> **Важно!**
>
> Для объектов **Задача** префикс номеров устанавливается подсистемой **Бизнес-процессы и задачи**. Использование подсистемы префиксации для объектов **Задача** будет вызывать ошибки в работе конфигурации.

В общем модуле `ПрефиксацияОбъектовСобытия` предусмотрено три процедуры – обработчика подписок, использование которых различается в зависимости от наличия в конфигурации справочника `Организации` и подсистемы **Обмен данными**:

- `УстановитьПрефиксОрганизации` – следует использовать в том случае, если требуется выполнять префиксацию объектов только в разрезе организаций.
- `УстановитьПрефиксИнформационнойБазы` – следует использовать в том случае, если используется подсистема **Обмен данными**, а префиксацию объектов в разрезе организаций выполнять не требуется. К таким типам объектов в основном относятся справочники и планы видов характеристик.
- `УстановитьПрефиксИнформационнойБазыИОрганизации` – следует использовать в том случае, если требуется выполнять префиксацию объектов в разрезе информационных баз и организаций одновременно. К таким типам объектов в основном относятся документы.

Разработчик может выбирать процедуры-обработчики в зависимости от требуемой бизнес-логики.

Если в конфигурации имеется справочник `Организации`, то для возможности префиксации объектов в разрезе организаций следует создать следующие объекты метаданных:

- реквизит `Префикс` (`Строка`, 2) у справочника `Организации`;
- параметр функциональной опции `Организация` с использованием справочника `Организации`;
- функциональная опция `ПрефиксыОрганизаций` с хранением в реквизите справочника `Организации`.

Создание подписок для переназначения номера (кода) объекта

Следует создать необходимое количество подписок на события для переназначения номера (кода) объекта при изменении даты или значения реквизита `Организация` в шапке объекта. После очистки объекту будет назначен новый код или номер.

Значения свойств подписок надо задать согласно таблице:

Имя свойства	Описание
`Имя`.	Имя следует задавать для удобства визуального восприятия подписки и с учетом ограничений, налагаемых на имена объектов метаданных, например `ПроверитьНомерДокументаПоДатеИОрганизации`.
`Источник`.	Источниками событий подписки могут быть только объекты типов: `Документы`, `Справочники`, `ПланыВидовХарактеристик`, `БизнесПроцессы`. В качестве источников для одной подписки могут быть использованы объекты только одного типа.
`Событие`.	`ПередЗаписью`.
`Обработчик`.	В качестве процедур – обработчиков подписок следует выбирать экспортные процедуры общего модуля `ПрефиксацияОбъектовСобытия`.

В общем модуле `ПрефиксацияОбъектовСобытия` предусмотрено пять процедур – обработчиков подписок, каждую из которых необходимо использовать исходя из типа объектов – источников подписок и наличия реквизита `Организация` в шапке этих объектов:

- `ПроверитьКодСправочникаПоОрганизации` – следует использовать для справочников, у которых нумерация кодов выполняется в разрезе справочника `Организации`.
- `ПроверитьНомерБизнесПроцессаПоДате` – следует использовать для бизнес-процессов, у которых нумерация выполняется только в пределах заданной периодичности.
- `ПроверитьНомерБизнесПроцессаПоДатеИОрганизации` – следует использовать для бизнес-процессов, у которых нумерация выполняется в пределах заданной периодичности и в разрезе справочника `Организации`.
- `ПроверитьНомерДокументаПоДате` – следует использовать для документов, у которых нумерация выполняется только в пределах заданной периодичности.
- `ПроверитьНомерДокументаПоДатеИОрганизации` – следует использовать для документов, у которых нумерация выполняется в пределах заданной периодичности и в разрезе справочника `Организации`.

В большинстве случаев в конфигурации требуется создание только трех подписок: двух для назначения префиксов справочникам и документам и одной для очистки номеров документов. Пример для создания таких подписок представлен в таблице.

Имя подписки	Обработчик
`УстановитьПрефиксИнформационнойБазыКодуСправочника`	`ПрефиксацияОбъектовСобытия.УстановитьПрефиксИнформационнойБазы`
`УстановитьПрефиксИнформационнойБазыИОрганизацииНомеруДокумента`	`ПрефиксацияОбъектовСобытия.УстановитьПрефиксИнформационнойБазыИОрганизации`
`ПроверитьНомерДокументаПоДатеИОрганизации`	`ПрефиксацияОбъектовСобытия.ПроверитьНомерДокументаПоДатеИОрганизации`

Если объекту требуется устанавливать префикс организации, то в составе реквизитов шапки объекта должен присутствовать реквизит `Организация`. Обязательные значения свойств реквизита указаны в таблице.

Свойство	Значение
`Имя`	`Организация`
`Тип`	`СправочникСсылка.Организации`

Настройка прав доступа пользователей

Настройка прав доступа пользователей к данным подсистемы Префиксация объектов не требуется.

Использование при разработке конфигурации

Формат префиксов и логика их установки объектам

Назначение префикса номеру или коду объекта происходит в момент первой записи объекта, если код или номер не были назначены вручную. Назначение префикса также происходит, если номер или код объекта были очищены перед его записью.

Префикс имеет фиксированную длину: 3 или 5 символов. В префиксе нельзя использовать символ «-» (дефис); он используется для разделения префикса с номером и добавляется автоматически. Соответственно, префикс формируется по одному из двух шаблонов: **ОРИБ-** или **ИБ-**, где ОР – 2 символа префикса справочника **Организации**; ИБ – 2 символа префикса информационной базы; «-» (дефис) – разделитель префикса.

Для конфигураций, в которых используется подсистема префиксации, длину номеров документов и кодов справочников рекомендуется устанавливать не менее 11 символов.

Если префикс информационной базы или префикс организации не указан, то считается, что такой префикс отсутствует, он заменяется двумя нулями – «00».

Для предопределенных элементов ссылочных типов данных, создаваемых в режиме конфигуратора, рекомендуется назначать префикс кода элемента в момент его создания. Как правило, для таких элементов требуется назначить только префикс информационной базы. Например, для предопределенного элемента справочника в конфигураторе может быть назначен такой код: «УТ-00000001». Если этого не выполнить, то, если настроен обмен данными между разными конфигурациями, для справочников с предопределенными элементами может быть нарушена уникальность кодов этих элементов.

Программный интерфейс

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Для взаимодействия с подсистемой печати предусмотрена экспортная функция общего модуля. Функция позволяет сформировать номер документа для вывода в печатной форме: `ПрефиксацияОбъектовСобытия.ПолучитьНомерНаПечать`.

Функция отбрасывает префикс организации, отбрасывает префикс информационной базы (опционально), отбрасывает пользовательские префиксы (опционально), удаляет лидирующие нули в номере объекта.

> Пример
>
> Документ имеет номер ФР0Г-А001234. После отбрасывания префикса организации и незначащих нулей номера для печати получим значение: Г-А1234.

В подсистеме предусмотрен обработчик события «При получении номера на печать». Обработчик срабатывает перед стандартной обработкой и предназначен для формирования номера объекта нестандартным способом. Например, когда формат номера документа отличается от стандартного. Для задания кода обработчика используется процедура `ПрефиксацияОбъектовКлиентСерверПереопределяемый.ПриПолученииНомераНаПечать`.

Дополнительные возможности по настройке формата префиксов

Подсистема предоставляет возможность добавлять к префиксу произвольную последовательность символов – например, дополнительный префикс «А» для документов **Счет-фактура на аванс**. Для этого в модуле объекта в обработчике `ПриУстановкеНовогоНомера` следует задать значение переменной `Префикс`.

Пример номера документа с установленным префиксом: ФР0Г-А001234 – документ № 1234 с пользовательским префиксом «А», заведенный на организацию «ФР», созданный в информационной базе «Г».

Переопределение имени реквизита, который содержит ссылку на организацию

В подсистеме предусмотрена стандартная обработка установки префикса организации. Стандартная обработка подразумевает, что значение ссылки на организацию выбирается из реквизита шапки объекта с именем **Организация**. Однако в некоторых объектах реквизит может именоваться иначе (например, `Владелец` или `ГоловнаяОрганизация`). Если ссылка на организацию в объекте располагается в реквизите с именем, отличным от стандартного имени `Организация`, то необходимо воспользоваться обработчиком `ПолучитьПрефиксообразующиеРеквизиты` в переопределяемом модуле подсистемы `ПрефиксацияОбъектовПереопределяемый`.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Проверка легальности получения обновления

Подсистема **Проверка легальности получения обновления** позволяет запросить у пользователя подтверждение, что файлы обновления были получены легальным способом. Может использоваться как перед обновлением информационной базы (после того как обновление уже было применено к базе, но до первого запуска) или непосредственно перед обновлением конфигурации средствами подсистемы **Обновление конфигурации**.

Настройка

Для настройки подсистемы в конфигурации дополнительных действий не требуется.

Настройка прав доступа пользователей

Для использования подсистемы достаточно одной роли:

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Открытие формы для подтверждения легальности получения обновления.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Профили безопасности

Подсистема **Профили безопасности** позволяет администратору автоматически настраивать профили безопасности информационной базы для используемых функций приложения (в зависимости от включенных функциональных опций, настроек синхронизации данных и т. п.). Подробнее о профилях безопасности см. документацию по платформе **1С:Предприятие**.

Использование при разработке конфигурации

В клиент-серверных базах администратор кластера может запретить приложению выполнять действия, которые могут быть потенциально опасны для функционирования. Для этого он может назначить информационной базе профиль безопасности и тогда потенциально опасная функциональность будет ограничена в тех пределах, которые описаны в этом профиле.

В профиль безопасности включаются разрешения на внешние ресурсы, необходимые для работоспособности функций приложения:

- Интернет-ресурсы, протоколы и порты подключения;
- каталоги файловой системы;
- внешние компоненты и COM-классы;
- запускаемые приложения операционной системы;

- подключенные расширения.

Для упрощения настройки профилей безопасности требуемые внешние ресурсы необходимо перечислить в процедуре `ПриЗапросеРазрешенийНаИспользованиеВнешнихРесурсов` общего модуля `РаботаВБезопасномРежимеПереопределяемый`. Это могут быть как статические ресурсы, безусловно необходимые для работы приложения всегда, так и ресурсы, востребованность которых определяется включением определенных функциональных опций, настройками интеграции и т.п.

При включении и выключении функции приложения, которая предполагает работу с внешними ресурсами, необходимо предварительно запрашивать у администратора разрешение на включение/выключение доступа к ним. Для этого следует вызвать процедуру `ПрименитьЗапросыНаИспользованиеВнешнихРесурсов` общего модуля `РаботаВБезопасномРежимеКлиент`. См. пример в демонстрационной конфигурации в обработке `_ДемоЗагрузкаНоменклатурыИзПрайсЛистаПрофилиБезопасности`, в которой выполняется запрос на разрешение доступа к Интернет-ресурсу или сетевому каталогу, с которого на сервере **1С:Предприятие** загружается файл с прайс-листом.

Для просмотра списка разрешенных внешних ресурсов в состав подсистемы также входит отчет **ИспользуемыеВнешниеРесурсы**.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме перечисленных:

- константа `АвтоматическиНастраиватьРазрешенияВПрофиляхБезопасности`,
- константа `ИспользуютсяПрофилиБезопасности`,
- константа `ПрофильБезопасностиИнформационнойБазы`,
- регистр сведений `ЗапросыРазрешенийНаИспользованиеВнешнихРесурсов`,
- регистр сведений `РазрешенияНаИспользованиеВнешнихРесурсов`,
- регистр сведений `РежимыПодключенияВнешнихМодулей`.

Работа в модели сервиса

Подсистема **Работа в модели** содержит базовый функционал, обязательный для всех прикладных решений, рассчитанных на работу в модели сервиса, а также ряд подсистем, которые расширяют другие подсистемы для работы в модели сервиса (например, **Работа с файлами в модели сервиса** и пр.).

Перед внедрением данной подсистемы рекомендуется ознакомиться с методикой разработки **1С:Технология разработки решений 1cFresh** и разделом Механизм разделения данных в книге **Руководство разработчика** из состава документации к платформе **1С:Предприятие**.

> **Важно!**
>
> В составе подсистемы поставляются два общих реквизита-разделителя: `ОбластьДанныхОсновныеДанные` и `ОбластьДанныхВспомогательныеДанные`. Корректное функционирование подсистем библиотеки при добавлении других разделителей в общем случае не поддерживается.
>
> **Примечание**
>
> Непосредственно в саму подсистему **Работа в модели сервиса** объекты библиотеки не включены, они относятся к дочерним подсистемам в ветке **Работа в модели сервиса**.

Настройка

Разработка форм настройки системы для администратора информационной базы и для администратора области данных

В отличие от локального режима работы, при работе в модели сервиса функции администрирования системы разделены на два уровня:

- Администратор информационной базы (роли `ПолныеПрава` и `АдминистраторСистемы`) выполняет настройку и обслуживание системы в целом, для всех областей данных. Например, настройку доступа к внешним ресурсам на сервере **1С:Предприятия** (прокси-сервер, почта, тома хранения файлов и т. п.), каких-либо общих параметров системы, которые должны распространяться на пользователей всех областей данных.
- Администратор области данных (роль `ПолныеПрава`) выполняет настройку параметров системы для конкретной области данных. Например: включение/отключение функциональных опций, ведение списка пользователей, удаление помеченных объектов и т. п.

Таким образом, при разработке форм настроек следует учитывать, что настройки информационной базы должны быть видны только администратору информационной базы (роль `АдминистраторСистемы`), а настройки области данных – всем администраторам (роль `ПолныеПрава`).

Если используется подсистема **Настройки программы**, то прикладные формы настроек рекомендуется разрабатывать в том же стиле – в виде панелей настроек, которые «подстраиваются» под права пользователя. Также необходимо отразить в документации тот факт, что для настройки общих параметров администратору информационной базы необходимо войти в любую область данных. Подробнее см. раздел документации Настройки программы.

В противном случае требуется разработать отдельные формы настроек для администратора информационной базы и для администратора области данных. Формы настроек должны хорошо подходить как для сценариев настройки двумя разными людьми при работе в модели сервиса, так и для сценария настройки одним человеком при работе в локальном режиме:

- Формы настроек, предназначенные для администратора информационной базы, должны быть доступны только роли `АдминистраторСистемы` и скрываться при работе в локальном режиме. Для этого форму следует включить в состав функциональной опции

`ИспользоватьРазделениеПоОбластямДанных`.

- Формы настроек, предназначенные для администратора области данных, должны содержать все настройки (и настройки информационной базы и настройки области данных), быть доступны для роли `ПолныеПрава` и рассчитаны на работу без роли `АдминистраторСистемы`.

Настройка общих реквизитов-разделителей и объектов метаданных

При работе в модели сервиса в конфигурации должен быть включен механизм разделения данных, который позволяет разделить все хранимые данные, а также работу прикладного решения на отдельные части. Для разделения данных в составе библиотеки поставляются два общих реквизита-разделителя:

- общий реквизит `ОбластьДанныхОсновныеДанные` – тип `Число(7)`, использование разделяемых данных – `Независимо`. Значение этого разделителя должно быть установлено для всех пользователей информационной базы (кроме администратора информационной базы);
- общий реквизит `ОбластьДанныхВспомогательныеДанные` – тип `Число(7)`, использование разделяемых данных – `Независимо и совместно`. Данный разделитель не используется для разделения пользователей и аутентификации.

При внедрении подсистемы **Работа в модели сервиса** в состав конфигурации важно сохранять исходный порядок сортировки общих реквизитов-разделителей в дереве метаданных.

При работе в модели сервиса все данные, хранящиеся в информационной базе (и объекты метаданных, соответствующие этим данным), могут быть разделены на следующие виды:

- **Общие (неразделенные) данные** – данные, которые являются общими для всех областей данных и не разделяются общими реквизитами-разделителями. Из сеансов, в которых не установлены значения разделителей, общие данные доступны только для чтения.
- **Разделенные данные** – данные, относящиеся к области данных, разделяются общими реквизитами-разделителями, поставляемыми в составе подсистемы **Работа в модели сервиса**:
  - **Основные данные** – данные области, обладающие прикладным характером. Объекты метаданных, соответствующие общим данным, разделяются разделителем `ОбластьДанныхОсновныеДанные`;
  - **Вспомогательные данные** – данные, логически являющиеся частью данных области, но доступные из неразделенных сеансов, разделяются разделителем `ОбластьДанныхВспомогательныеДанные`.

По умолчанию большинство объектов метаданных должны быть включены в состав общего реквизита `ОбластьДанныхОсновныеДанные`, т. е. должны быть разделенными. Поскольку у общего реквизита `ОбластьДанныхОсновныеДанные` свойство `Автоиспользование` установлено в значение `Использовать`, то для вновь создаваемых объектов конфигурации ничего предпринимать не требуется.

Из состава разделителя `ОбластьДанныхОсновныеДанные` должны быть удалены следующие объекты:

- Все регламентные задания.
- Объекты метаданных, предназначенные для хранения настроек информационной базы (общих для всех областей данных). В основном это константы, например: `МинимальныйИнтервалРегламентныхЗаданийДОИОВМоделиСервиса`.
- Различные классификаторы, общие для всех областей данных, такие как классификатор валют, адресный классификатор и т. п.

В состав разделителя `ОбластьДанныхВспомогательныеДанные` должны включаться только служебные данные (не имеющие ценности для абонента). Для общего реквизита свойство `Автоиспользование` установлено в значение **Не использовать** – таким образом вновь создаваемые объекты метаданных не будут включаться в состав данного разделителя.

Все объекты метаданных, относящиеся к общим (неразделенным) данным, должны быть включены в состав любой из подписок на события, для которой в качестве обработчика назначена одна из следующих процедур (в зависимости от типа объекта метаданных):

- `РаботаВМоделиСервисаБСП.КонтрольНеразделенныхОбъектовПриЗаписи`,
- `РаботаВМоделиСервисаБСП.КонтрольНеразделенныхНаборовЗаписейПриЗаписи`.

Например, это могут быть следующие подписки на события, поставляемые непосредственно в составе подсистемы `БазоваяФункциональностьВМоделиСервиса`:

- `КонтрольНеразделенныхОбъектовПриЗаписи`,
- `КонтрольНеразделенныхНаборовЗаписейПриЗаписи`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Функции администратора области данных: настройка параметров системы для конкретной области данных и неограниченный доступ ко всем разделенным данным.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор области данных	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Ответственный за нормативно-справочную информацию (в модели сервиса)	• `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеДополнительныхРеквизитовИСведений`, • `ДобавлениеИзменениеКурсовВалют`, • `ДобавлениеИзменениеГрафиковРаботы`, • `ДобавлениеИзменениеВидовКонтактнойИнформации`.

Использование при разработке конфигурации

Работа при условно выключенном разделении

Независимо от того, рассчитана ли конфигурация на работу в локальном режиме, ее код должен быть работоспособен при условно выключенном разделении (т. е. в «обычном» режиме, как будто никакого разделения нет). Поэтому если в конфигурации есть какой-то код, специфичный для модели сервиса, то его выполнение необходимо предварять проверкой того, что разделение включено.

Для проверки того, что разделение данных включено, следует использовать функцию `РазделениеВключено` общего модуля `ОбщегоНазначения`.

Получение списка пользователей для отображения

Для отображения списка пользователей в большинстве случаев следует использовать формы справочника **Пользователи**. В тех случаях, когда это невозможно, при включенном разделении данных следует скрывать из списка неразделенных пользователей. Пример реализации см. в форме списка справочника **Пользователи**.

Роли разделенных пользователей

При разработке ролей, которые предполагается использовать для работы разделенных пользователей, следует иметь в виду следующее:

- Для всех объектов метаданных, не входящих в состав разделителей `ОбластьДанныхОсновныеДанные` и `ОбластьДанныхВспомогательныеДанные`, должны быть сняты права `Добавление`, `Изменение`, `Удаление`.
- А также должны быть сняты права на конфигурацию:
  - администрирование,
  - обновление конфигурации базы данных,
  - монопольный режим,
  - толстый клиент,
  - внешнее соединение,
  - automation,
  - интерактивное открытие внешних обработок,
  - интерактивное открытие внешних отчетов,
  - режим **Все функции**.

При работе в разделенном режиме роли, которые не удовлетворяют вышеприведенным требованиям, будут автоматически удалены из состава профилей групп доступа и будут недоступны для назначения пользователям. Кроме того, при попытке входа в систему пользователя с такими ролями будет выдана ошибка и работа будет завершена.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме следующих:

- константа `ВнутреннийАдресМенеджераСервиса`,
- константа `ВыполнитьРезервноеКопированиеОбластиДанных`,
- константа `ДатаПоследнегоСтартаКлиентскогоСеанса`,
- константа `ИспользованиеКаталогаДополнительныхОтчетовИОбработокВМоделиСервиса`,
- константа `КаталогОбменаФайламиВМоделиСервиса`,
- константа `КаталогОбменаФайламиВМоделиСервисаLinux`,
- константа `КлючОбластиДанных`,
- константа `КонечнаяТочкаМенеджераСервиса`,
- константа `КопироватьОбластиДанныхИзЭталонной`,
- константа `МаксимальнаяДлительностьВыполненияИсполняющегоФоновогоЗадания`,
- константа `МаксимальноеКоличествоИсполняющихФоновыхЗаданий`,
- константа `МинимальныйИнтервалРегламентныхЗаданийДополнительныхОтчетовИОбработокВМоделиСервиса`,
- константа `НезависимоеИспользованиеДополнительныхОтчетовИОбработокВМоделиСервиса`,
- константа `ПоддержкаРезервногоКопирования`,
- константа `ПредставлениеОбластиДанных`,
- константа `ПрефиксОбластиДанных`,
- константа `ПриоритетОбновленияВОбластяхДанных`,

- константа `РазмерБлокаПередачиФайла`,
- константа `РазрешитьВыполнениеДополнительныхОтчетовИОбработокРегламентнымиЗаданиямиВМоделиСервиса`,
- константа `РежимИспользованияИнформационнойБазы`,
- константа `СообщениеБлокировкиПриОбновленииКонфигурации`,
- константа `ЧасовойПоясОбластиДанных`,
- справочник `ОчередьЗаданий`,
- справочник `ОчередьЗаданийОбластейДанных`,
- справочник `ПоставляемыеДанные`,
- справочник `ПоставляемыеДополнительныеОтчетыИОбработки`,
- справочник `СообщенияОбластейДанных`,
- справочник `ШаблоныЗаданийОчереди`,
- регистр сведений `ВерсииПодсистемОбластейДанных`,
- регистр сведений `ИспользованиеПоставляемыхДополнительныхОтчетовИОбработокВОбластяхДанных`,
- регистр сведений `ИспользованиеДополнительныхОтчетовИОбработокСервисаВАвтономномРабочемМесте`,
- регистр сведений `ОбластиДанных`,
- регистр сведений `ОбластиТребующиеОбработкиПоставляемыхДанных`,
- регистр сведений `ОчередьИзвлеченияТекста`,
- регистр сведений `ОчередьИнсталляцииПоставляемыхДополнительныхОтчетовИОбработокВОбластиДанных`,
- регистр сведений `ПоставляемыеДанныеТребующиеОбработки`,
- регистр сведений `РейтингАктивностиОбластейДанных`,
- регистр сведений `УдалитьРейтингАктивностиОбластейДанных`.

Работа с почтовыми сообщениями

Подсистема **Работа с почтовыми сообщениями** добавляет в конфигурацию программный интерфейс по отправке сообщений электронной почты, а также пользовательский интерфейс для поддержки учетных записей электронной почты.

Настройка

В случае если в конфигурации не используется подсистема **Настройки программы**, в рабочем месте администратора приложения необходимо разместить справочник `УчетныеЗаписиЭлектроннойПочты` и общую команду `НастройкаСистемнойУчетнойЗаписиЭлектроннойПочты`. См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

Особые случаи внедрения подсистемы

- Внедрение подсистем «Работа с почтовыми сообщениями» и «Управление доступом».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Работа с почтовыми сообщениями** следует использовать роли:

№	Роли и их назначение
1.	• `ЧтениеУчетныхЗаписейЭлектроннойПочты` • Отправка и получение электронных писем с разрешенных учетных записей электронной почты без возможности добавления персональных учетных записей. • При совместном использовании с подсистемой **Управление доступом** для ограничения доступа к общим учетным записям электронной почты используется вид доступа `УчетныеЗаписиЭлектроннойПочты`.
2.	`ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты` Отправка и получение электронных писем с разрешенных учетных записей электронной почты, а также добавление персональных учетных записей. • При совместном использовании с подсистемой **Управление доступом** для ограничения доступа к общим учетным записям электронной почты используется вид доступа `УчетныеЗаписиЭлектроннойПочты`.
3.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Добавление и изменение учетных записей электронной почты. Удаление помеченных на удаление объектов подсистемы.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Пользователь: • Отправка и получение электронных писем с общих и персональных учетных записей электронной почты	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты`.
2.	Администратор: • Добавление новых и изменение существующих учетных записей электронной почты. Ограничение: изменение персональной почты пользователя требует ввода пароля. • Управление доступом пользователей к общим учетным записям	• **Полные права** (из подсистемы **Базовая функциональность**)

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включить в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Работа с файлами

Подсистема **Работа с файлами** предназначена для коллективного редактирования файлов произвольного формата в иерархической структуре папок. Хранение файлов может быть организовано как непосредственно в информационной базе, так и внешним образом – в томах (сетевых ресурсах). Вместе с файлом может быть сохранена и история его изменений (версии файла).

Поддерживаются такие возможности работы с файлами, как добавление файлов в базу из файловой системы, создание файлов по шаблону, электронная подпись и шифрование, поддержка группового изменения и полнотекстового поиска по файлам.

Кроме того, имеется возможность присоединять файлы к произвольным объектам конфигурации ссылочного типа.

> **Важно!**
>
> При работе в веб-клиенте имеется ограничение: в веб-браузерах Chrome и Firefox подсистема работает без установленного расширения для работы с 1С:Предприятием.

Настройка

Размещение в командном интерфейсе

В случае если в конфигурации не используется подсистема **Настройки программы**, на рабочем месте администратора приложения необходимо разместить константы:

- `ЗапрещатьЗагрузкуФайловПоРасширению`,
- `ИзвлекатьТекстыФайловНаСервере`,
- `МаксимальныйРазмерФайла`,
- `СписокЗапрещенныхРасширений`,
- `СписокРасширенийТекстовыхФайлов`,
- `СписокРасширенийФайловOpenDocument`,
- `ХранитьФайлыВТомахНаДиске`.

См. пример в демонстрационной конфигурации в форме `НастройкиРаботыСФайлами` обработки `ПанельАдминистрированияБСП`.

Затем разместить в командном интерфейсе администратора приложения следующие объекты метаданных:

- справочник `ТомаХраненияФайлов`,
- обработка `АвтоматическоеИзвлечениеТекстов`.

Для доступа к списку файлов в томе необходимо в критерии отбора `ФайлыВТоме` установить флажок **Использовать стандартные команды**.

Также необходимо разместить в командном интерфейсе администратора команды справочника **Файлы**:

- Команда **Файлы** (имя `ПапкиФайлов`) – открывает форму дерева папок с отображением файлов в выбранной папке.
- Команда **Все файлы** – отображает все файлы, которые содержатся в справочнике.
- Команда **Редактируемые файлы** (имя `ЗанятыеФайлы`) – отображает все редактируемые файлы, которые содержатся в справочнике.

См. пример в демонстрационной конфигурации в подразделе **Работа с файлами** раздела `Органайзер` (подсистема `_ДемоРаботаСФайлами` подсистемы `_ДемоОрганайзер`).

Для редактирования персональных настроек работы с файлами следует предусмотреть в конфигурации форму настройки и разместить на ней следующие элементы:

- Действие при выборе файла.
- Выбор режима открытия при выборе файла.
- Показывать подсказки при редактировании файлов (только веб-клиент).
- Показывать занятые файлы при завершении работы.
- Показывать колонку **Размер** в списках файлов.
- Сравнивать версии при помощи.
- Настройка основного рабочего каталога.
- Настройка сканирования.
- Установить расширение для работы с 1С:Предприятием.

Пример реализации персональных настроек работы с файлами можно найти на закладке **Работа с файлами** общей формы `_ДемоМоиНастройки`.

Настройка присоединения файлов к объектам

Необходимо принять решение по поводу состава объектов конфигурации, которые должны содержать присоединенные файлы («объекты с файлами»). Такими объектами, как правило, являются справочники или документы, при которых могут содержаться файлы-вложения. Примеры: справочник **Товары** с присоединенными к нему файлами-спецификациями, сертификатами и т. п.; справочник **Должности** с приложенными к нему текстами должностных обязанностей, требованиями к соискателю и т. п.

В зависимости от того, используются ли в конфигурации ограничения доступа на уровне записей (RLS; см. раздел Управление доступом) и их разнообразия, принять решение по варианту хранения присоединенных файлов:

- В большинстве случаев для каждого «объекта с файлами» (типа владельца) следует создать отдельный справочник присоединенных файлов. Данный вариант рекомендуется, если в конфигурации предусмотрены разные права доступа к владельцам.
  - Например, если в конфигурации для справочников `Партнеры` и `Проекты` предусмотрена разная логика RLS, то с помощью RLS к отдельным справочникам присоединенных файлов `ПартнерыПрисоединенныеФайлы` и `ПроектыПрисоединенныеФайлы` можно обеспечить, чтобы пользователи просматривали и редактировали присоединенные файлы только у доступных им партнеров и проектов.
  - Если же в конфигурации отсутствует RLS, то аналогичным образом права на справочники присоединенных файлов синхронно выдаются с правами на их владельцев.
- В некоторых случаях достаточно одного справочника присоединенных файлов сразу для нескольких «объектов с файлами» (типов владельцев). Этот вариант подходит, если владельцев немного (не более 10), а права доступа к владельцам полностью совпадают и с высокой вероятностью не будут пересматриваться в дальнейшем.
  - Например, в одном справочнике присоединенных файлов можно реализовать хранение файлов, присоединенных к различным документам продажи (**Счета покупателям**, **Реализация** и т. п.), RLS к которым полностью одинаков. При этом для корректного ограничения доступа к присоединенным файлам в зависимости от типа их владельца в RLS к справочнику присоединенных файлов также следует задействовать специальные виды доступа `ПравоЧтения` и `ПравоИзменения` (при совместном внедрении с подсистемой **Управление доступом**).
- Также допустимо не создавать справочники присоединенных файлов, а выбрать вариант хранения в имеющемся справочнике `Файлы` в тех случаях, когда в конфигурации отсутствует логика RLS или когда она есть, но в «объектах с файлами» (типах владельцев) предусмотрена запись наборов значений доступа.
  - В первом случае доступ к справочнику `Файлы`, как и для владельцев, предоставляется без ограничений (но при совместном внедрении с подсистемой **Управление доступом** остается базовое ограничение по типам владельцев).
  - Во втором – доступ к справочнику Файлы будет обеспечен «как у владельца» с помощью стандартного шаблона `#ПоЗначениямИНаборамРасширенный` (подробнее см. раздел Управление доступом).

В случае отдельного справочника для хранения файлов

Для каждого «объекта с файлами» (типа владельца), использующего отдельный справочник для хранения файлов, выполнить настройку:

1. Создать справочник для хранения присоединенных файлов. Для этого в качестве заготовки скопировать в конфигурацию справочник `_ДемоПроектыПрисоединенныеФайлы` из демонстрационной конфигурации и задать ему имя по шаблону: `<Префикс>ПрисоединенныеФайлы`, где `Префикс` – имя объекта метаданных, для которого настраиваются присоединенные файлы. Например, для справочника `Номенклатура` справочник с файлами должен называться `НоменклатураПрисоединенныеФайлы`. Задать синоним, например: **Присоединенные файлы** (**Номенклатура**).
2. У реквизита `ВладелецФайла` установить «объект с файлами» (тип владельца). Например, `СправочникСсылка.Номенклатура`.
3. Принять решение о возможности создавать группы в списках присоединенных файлов. Для обеспечения возможности создания папок необходимо включить у справочника свойство `Иерархический`, установить **Вид иерархии**: **Иерархия групп и элементов**. Изменить свойства `Использование`: **Для группы и элемента** реквизитов `Автор`, `ВладелецФайла`, `ДатаМодификацииУниверсальная`, `ДатаСоздания`, `Изменил`, `ИндексКартинки`, `Описание`.
4. Опционально. Добавить реквизит `Служебный` типа `Булево`, если в справочнике хранятся присоединенные файлы, которые содержат служебную информацию и должны быть скрыты от пользователей.
5. Включить в состав определяемых типов `ПрисоединенныйФайл` (ссылки) и `ПрисоединенныйФайлОбъект` (объекты) справочник, созданный на шаге 1. Например, `НоменклатураПрисоединенныеФайлы`.
6. Включить в состав плана обмена `ОбновлениеИнформационнойБазы` справочник, созданный на шаге 1.
7. Создать подписки на события, включив в их состав типов свойства `Источник` справочник с файлами, созданный на шаге 1 (например, `СправочникМенеджер.НоменклатураПрисоединенныеФайлы`):

Подписка	Событие	Обработчик
`ОпределитьФормуПрисоединенногоФайла`	`ОбработкаПолученияФормы`	`РаботаСФайламиКлиентСервер.ОпределитьФормуПрисоединенногоФайла`
`УстановитьПометкуУдаленияПрисоединенныхФайловДокументов`	`ПередЗаписью`	`РаботаСФайлами.УстановитьПометкуУдаленияПрисоединенныхФайловДо`

См. примеры подписок в демонстрационной конфигурации `_ДемоОпределитьФормуПрисоединенногоФайла` и `_ДемоУстановитьПометкуУдаленияПрисоединенныхФайловДокументов`.

В случае справочника Файлы для хранения файлов

Для каждого «объекта с файлами» (типа владельца), использующего справочник `Файлы` для хранения файлов, выполнить настройку: включить в состав определяемого типа `ВладелецФайлов` (ссылки) владельцев. Например, `ДокументСсылка.ЗаказПокупателя`.

Для корректной работы RLS права на «объект с файлами» и справочник `Файлы` должны предоставляться одной группой доступа.

Например, если права на документ `ЗаказПокупателя` предоставляются группой доступа `Добавление и изменение заказов покупателей`, а права на справочник `Файлы` предоставляются группой доступа `Добавление и изменение папок и файлов`, то пользователь не увидит файлов, присоединенных к документу `ЗаказПокупателя`.

Для корректного предоставления прав необходимо создать новую группу доступа, в которую будут добавлены права на просмотр документа `ЗаказПокупателя` и роль `Добавление и изменение папок и файлов`.

В обоих случаях

Для каждого «объекта с файлами» (типа владельца), использующего любой справочник для хранения файлов, выполнить настройку:

1. Расширить состав определяемых типов `ВладелецПрисоединенныхФайлов` (ссылки) и `ВладелецПрисоединенныхФайловОбъект` (объекты, кроме документов), добавив в него тип «объект с файлами». Например `СправочникСсылка.Номенклатура`, `ДокументСсылка.СчетНаОплатуПокупателю`, `ДокументСсылка.ЗаказПокупателя`.
2. Если «объект с файлами» это документ, расширить состав типов свойства `Источник` подписки `УстановитьПометкуУдаленияПрисоединенныхФайловДокументов`, включив в него тип документа. Например, `ДокументОбъект.СчетНаОплатуПокупателю`, `ДокументОбъект.ЗаказПокупателя`.
3. Если при интерактивном копировании объекта, содержащего присоединенные файлы, требуется автоматическое копирование файлов в новый объект, то в форме объекта необходимо:
   - в модуле формы в процедуре `ПриСозданииНаСервере` при вызове процедуры `ПриСозданииНаСервере` общего модуля `РаботаСФайлами` передать третий параметр:

```
НастройкиРаботыСФайламиВФорме = РаботаСФайлами.НастройкиРаботыСФайламиВФорме();
НастройкиРаботыСФайламиВФорме.КопироватьПрисоединенныеФайлы = Истина;
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ДобавляемыеЭлементы, НастройкиРаботыСФайламиВФорме);
```

   - в модуле формы в процедуру `ПриЗаписиНаСервере` вставить следующий код:

```
РаботаСФайлами.ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи, ЭтотОбъект);
```

4. Если перед закрытием формы объекта-владельца файлов требуется проверить наличие занятых текущим пользователем файлов, то необходимо:
   - Добавить реквизит формы `МожноЗакрытьФормуСФайлами` типа `Булево`.
   - В обработчике формы `ПередЗакрытием` вставить вызов процедуры `ПоказатьПодтверждениеЗакрытияФормыСФайлами` общего модуля `РаботаСФайламиКлиент` по шаблону:

```
&НаКлиенте
Процедура ПередЗакрытием(Отказ, ЗавершениеРаботы, ТекстПредупреждения, СтандартнаяОбработка)
    РаботаСФайламиКлиент.ПоказатьПодтверждениеЗакрытияФормыСФайлами(ЭтотОбъект, Отказ, ЗавершениеРаботы, Объект.Ссылка);
КонецПроцедуры
```

   - Другой код, который необходимо выполнять в обработчике `ПередЗакрытием`, следует размещать после вызова процедуры и проверки параметра `Отказ`. Например:

```
&НаКлиенте
Процедура ПередЗакрытием(Отказ, ЗавершениеРаботы, ТекстПредупреждения, СтандартнаяОбработка)
    РаботаСФайламиКлиент.ПоказатьПодтверждениеЗакрытияФормыСФайлами(ЭтотОбъект, Отказ, ЗавершениеРаботы, Объект.Ссылка);
    Если Отказ Тогда
        Возврат;
    КонецЕсли;
    <другой код...>
КонецПроцедуры
```

Особые случаи внедрения подсистемы

- Внедрение подсистем «Работа с файлами» и «Управление доступом».
- Внедрение подсистемы «Работа с файлами» без подсистемы «Свойства».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Работа с файлами** следует использовать следующие роли:

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Настройка подсистемы, создание томов для хранения файлов, удаление помеченных на удаление файлов.
2.	`БазовыеПраваБСП` Просмотр общих форм.
3.	`ДобавлениеИзменениеПапокИФайлов` Просмотр, добавление, редактирование файлов в разрешенных папках (справочники **Файлы** и **Папки файлов**). Права на каждую папку файлов настраиваются отдельно.
4.	`ДобавлениеИзменениеПапокИФайловВнешнимиПользователями` Просмотр, добавление, редактирование файлов в разрешенных папках для внешних пользователей (справочники **Файлы** и **Папки файлов**). Права на каждую папку файлов настраиваются отдельно.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для добавления/изменения и чтения «объектов с файлами». В этом случае настройка доступа к справочнику присоединенных файлов устанавливается аналогично «объекту с файлами».

№	Вспомогательные роли и их назначение
1.	`<ДобавлениеИзменениеНоменклатуры>` Ведение списка объектов с файлами, добавление и изменение присоединенных к ним файлов, подписание и шифрование файлов.
2.	`<ЧтениеНоменклатуры>` Просмотр объектов с файлами, присоединенных к ним файлов, открытие файлов во внешнем приложении.

Пример настройки прав доступа пользователей приведен ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	**Администратор** Общее администрирование системы	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	**Менеджер** Добавление, изменение «объектов с файлами» и их файлов. Редактирование файлов, подписание файлов, шифрование, обновление данных файла из другого файла	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеНоменклатуры` (из прикладной конфигурации)
3.	**Пользователь** Просмотр списка файлов, открытие карточки файла, открытие файла для просмотра	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**), • `ЧтениеНоменклатуры` (из прикладной конфигурации)

Использование при разработке конфигурации

Создание реквизитов типа «присоединенный файл»

При необходимости можно добавить к произвольному объекту ссылочного типа реквизит типа `СправочникСсылка.Файлы` или собственного справочника присоединенных файлов. Например, реквизит `ПрисоединенныйФайл` или `Фотография` в справочнике физических лиц. Использование этого реквизита в пользовательском интерфейсе определяется по месту и может быть, например, таким:

- Гиперссылка в форме объекта, при нажатии на которую открывается карточка файла или сам файл.
- Поле картинки в режиме гиперссылки, при нажатии на которую открывается карточка файла.

Управление присоединенными файлами из формы «объекта с файлами»

В форме любого «объекта с файлами» можно подключить настраиваемые элементы управления присоединенными файлами. Предусмотрены следующие варианты использования в пользовательском интерфейсе:

- Гиперссылка для быстрого перехода к списку присоединенных файлов. Имеет настраиваемые заголовок и местоположение на форме, а также позволяет присоединять файлы из формы размещения.
- Поле реквизита типа «присоединенный файл». Предоставляет возможность интерактивной работы с присоединенным файлом без перехода к карточке файла.

Для размещения элементов управления на форме, необходимо:

- В модуле формы объекта в процедуру `ПриСозданииНаСервере` добавить фрагмент:
  - Для размещения на форме гиперссылки:

```
// СтандартныеПодсистемы.РаботаСФайлами
ПараметрыГиперссылки = РаботаСФайлами.ГиперссылкаФайлов();
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыГиперссылки);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  - Для размещения на форме поля реквизита:

```
// СтандартныеПодсистемы.РаботаСФайлами
ПараметрыПоля = РаботаСФайлами.ПолеФайла();
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыПоля);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  - Для размещения на форме комбинации элементов управления или нескольких полей реквизитов, элементы управления объединяются в массив:

```
// СтандартныеПодсистемы.РаботаСФайлами
ПараметрыГиперссылки = РаботаСФайлами.ГиперссылкаФайлов();
ПараметрыПоля = РаботаСФайлами.ПолеФайла();
ДобавляемыеЭлементы = Новый Массив;
ДобавляемыеЭлементы.Добавить(ПараметрыГиперссылки);
ДобавляемыеЭлементы.Добавить(ПараметрыПоля);
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ДобавляемыеЭлементы);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

- Для снижения времени открытия формы рекомендуется предварительно создать служебные реквизиты и группу формы, в которой будут размещены элементы:
  - `ПараметрыРаботыСФайлами` типа `Произвольный`;
  - `ИзображениеНаФорме` типа `Строка` - для размещения поля изображения на форме. Имя этого реквизита также указать в явном виде в коде:

```
// СтандартныеПодсистемы.РаботаСФайлами
ПараметрыПоля = РаботаСФайлами.ПолеФайла();
ПараметрыПоля.ПутьКДаннымИзображения = "ИзображениеНаФорме";
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыПоля);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  - `ГруппаРазмещения` типа `ГруппаФормы` - для размещения создаваемых элементов на форме. Имя группы также указать в явном виде в коде:

```
// СтандартныеПодсистемы.РаботаСФайлами
ПараметрыГиперссылки = РаботаСФайлами.ГиперссылкаФайлов();
ПараметрыГиперссылки.Размещение = "ГруппаРазмещения";
РаботаСФайлами.ПриСозданииНаСервере(ЭтотОбъект, ПараметрыГиперссылки);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  В ином случае реквизиты и элементы формы будут созданы программно, что замедлит открытие формы.
  - В модуле формы объекта в процедуру `ПриОткрытии` добавить фрагмент:

```
// СтандартныеПодсистемы.РаботаСФайлами
РаботаСФайламиКлиент.ПриОткрытии(ЭтотОбъект, Отказ);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  - В модуле формы объекта в процедуру `ОбработкаОповещения` добавить фрагмент:

```
// СтандартныеПодсистемы.РаботаСФайлами
РаботаСФайламиКлиент.ОбработкаОповещения(ЭтотОбъект, ИмяСобытия);
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

  - В область `ОбработчикиСобытийЭлементовШапкиФормы` модуля формы объекта поместить следующий код:

```
// СтандартныеПодсистемы.РаботаСФайлами
&НаКлиенте
Процедура Подключаемый_ПолеПредпросмотраНажатие(Элемент, СтандартнаяОбработка)

    РаботаСФайламиКлиент.ПолеПредпросмотраНажатие(ЭтотОбъект, Элемент, СтандартнаяОбработка);

КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ПолеПредпросмотраПеретаскивание(Элемент, ПараметрыПеретаскивания, СтандартнаяОбработка)

    РаботаСФайламиКлиент.ПолеПредпросмотраПеретаскивание(ЭтотОбъект, Элемент,
                ПараметрыПеретаскивания, СтандартнаяОбработка);

КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ПолеПредпросмотраПроверкаПеретаскивания(Элемент, ПараметрыПеретаскивания, СтандартнаяОбработка)

    РаботаСФайламиКлиент.ПолеПредпросмотраПроверкаПеретаскивания(ЭтотОбъект, Элемент,
                ПараметрыПеретаскивания, СтандартнаяОбработка);

КонецПроцедуры
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

- В область `ОбработчикиКомандФормы` модуля формы объекта поместить следующий код:

```
// СтандартныеПодсистемы.РаботаСФайлами
&НаКлиенте
Процедура Подключаемый_КомандаПанелиПрисоединенныхФайлов(Команда)
    РаботаСФайламиКлиент.КомандаУправленияПрисоединеннымиФайлами(ЭтотОбъект, Команда);
КонецПроцедуры
// Конец СтандартныеПодсистемы.РаботаСФайлами
```

См. также примеры в демонстрационной базе: в справочниках `_ДемоНоменклатура` и `_ДемоФизическиеЛица`.

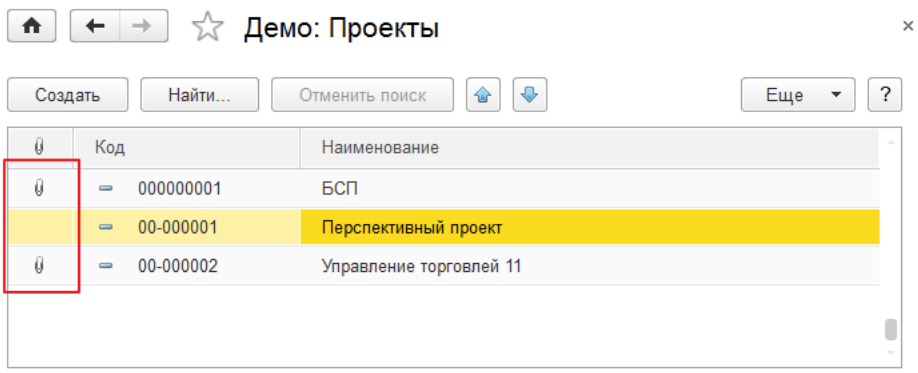
Признак наличия присоединенных файлов в списках

Если в списке «объектов с файлами» требуется вывести признак наличия присоединенных к объекту файлов, то необходимо в запросе динамического списка сделать левое соединение с регистром сведений `НаличиеФайлов` (по измерению `ОбъектСФайлами`).

Пример запроса из демонстрационной конфигурации (форма списка справочника `_ДемоПроекты`):

```
ВЫБРАТЬ
    Справочник_ДемоПроекты.Ссылка,
    Справочник_ДемоПроекты.ПометкаУдаления,
    Справочник_ДемоПроекты.Предопределенный,
    Справочник_ДемоПроекты.Код,
    Справочник_ДемоПроекты.Наименование,
    Справочник_ДемоПроекты.РеквизитДопУпорядочивания,
    ВЫБОР
        КОГДА НаличиеФайлов.ЕстьФайлы ЕСТЬ NULL ТОГДА
            0
        КОГДА НаличиеФайлов.ЕстьФайлы ТОГДА
            1
        ИНАЧЕ
            0
    КОНЕЦ КАК ЕстьФайлы
ИЗ
    Справочник._ДемоПроекты КАК Справочник_ДемоПроекты
        ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.НаличиеФайлов КАК НаличиеФайлов
        ПО Справочник_ДемоПроекты.Ссылка = НаличиеФайлов.ОбъектСФайлами
```

Пример списка «со скрепкой»:



Сканирование документов и фотографий

С помощью программного интерфейса подсистемы можно запускать сканирование документов и фотографий на клиентском компьютере, а также объединять изображения в многостраничные документы TIFF и PDF. Получать результат сканирования можно в виде присоединенного файла, двоичных данных или записать изображение в файл. См. процедуры `ДобавитьСоСканера`, `ДоступнаКомандаСканировать` и функцию `ДоступноСканирование` общего модуля `РаботаСФайламиКлиент`.

Сканирование возможно с любых устройств, драйвера которых поддерживают одну из технологий **SANE**, **WIA**, **TWAIN**, на операционных системах **Microsoft Windows** и **Linux**. Подробнее о версиях операционных систем см. Системные требования «1С:Предприятия 8».

Пример сканирования изображений в команде **Сканировать** в обработке **Сканирование**:

```
&НаКлиенте
Процедура ОбработкаКоманды(ПараметрКоманды, ПараметрыВыполненияКоманды)
    Если РаботаСФайламиКлиент.ДоступноСканирование() Тогда
        ПараметрыДобавленияСоСканера = РаботаСФайламиКлиент.ПараметрыДобавленияСоСканера();
        ПараметрыДобавленияСоСканера.ФормаВладелец = ЭтотОбъект;
        ПараметрыДобавленияСоСканера.ОбработчикРезультата = Новый ОписаниеОповещения("ОтсканироватьЛистЗавершение", ЭтотОбъект);
        ПараметрыДобавленияСоСканера.ТипРезультата = РаботаСФайламиКлиент.ТипРезультатаКонвертацииИмяФайла();
        ПараметрыДобавленияСоСканера.ТолькоОдинФайл = Истина;
        РаботаСФайламиКлиент.ДобавитьСоСканера(ПараметрыДобавленияСоСканера);
    КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ОтсканироватьЛистЗавершение(Результат, Контекст) Экспорт
    Если Результат <> Неопределено Тогда
        ФайловаяСистемаКлиент.ОткрытьФайл(Результат.ИмяФайла);
    КонецЕсли;
КонецПроцедуры
```

Пример формирование многостраничного PDF-файла с изображениями номенклатуры см. в справочнике **Демо: Номенклатура**, команда **Альбом изображений в PDF**:

```
&НаКлиенте
Процедура АльбомИзображенийВPDF(Команда)
    Изображения = ПрисоединенныеИзображения(Объект.Ссылка);
    ОписаниеОповещения = Новый ОписаниеОповещения("АльбомИзображенийВPDFЗавершение", ЭтотОбъект);
    ПараметрыКонвертацииГрафическогоДокумента = РаботаСФайламиКлиент.ПараметрыКонвертацииГрафическогоДокумента();
    РаботаСФайламиКлиент.ОбъединитьВМногостраничныйФайл(ОписаниеОповещения, Изображения, ПараметрыКонвертацииГрафическогоДокумента);
КонецПроцедуры

&НаКлиенте
Процедура АльбомИзображенийВPDFЗавершение(Результат, ДополнительныеПараметры) Экспорт
    Если Результат.Успешно Тогда
        ФайловаяСистемаКлиент.ОткрытьФайл(Результат.ИмяФайлаРезультата);
    КонецЕсли;
КонецПроцедуры
```

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме:

- константы `ИзвлекатьТекстыФайловНаСервере`,
- константы `ПараметрыХраненияФайловВИБ`,
- константы `ПутьКТомуБезУчетаРегиональныхНастроек`,
- константы `РежимОчисткиФайлов`,
- константы `СинхронизироватьФайлы`,
- константы `СоздаватьПодкаталогиСИменамиВладельцев`,
- константы `СпособХраненияФайлов`,
- константы `ХранитьФайлыВТомахНаДиске`,
- справочника `ТомаХраненияФайлов`.
- справочника `УчетныеЗаписиСинхронизацииФайлов`,
- регистра сведений `НастройкиСинхронизацииФайлов`,
- регистра сведений `НомераОтсканированныхФайлов`,
- регистра сведений `РабочиеКаталогиФайлов`,
- регистра сведений `СтатусыСинхронизацииФайловСОблачнымСервисом`,
- регистра сведений `ФайлыВРабочемКаталоге`.

Справочник `ХранилищеДвоичныхДанных` и регистры сведений `ХранилищеФайлов`, `УдалитьДвоичныеДанныеФайлов` нужно включать только в начальный образ подчиненного узла в РИБ (см. раздел Особенности создания начального образа подчиненного узла распределенной ИБ).

Если подсистема используется совместно с подсистемой **Обмен данными**, то для корректного создания начального образа с файлами, хранящимися в томах на компьютере, необходимо:

1. В модуле менеджера плана обмена (РИБ) в экспортной процедуре `ПриПолученииОписанияВариантаНастройки` указать значение свойства `ИмяФормыСозданияНачальногоОбраза`:

```
Процедура ПриПолученииОписанияВариантаНастройки(ОписаниеВарианта, ИдентификаторНастройки, ПараметрыКонтекста) Экспорт
    ...
    ОписаниеВарианта.ИмяФормыСозданияНачальногоОбраза = "ОбщаяФорма.СозданиеНачальногоОбразаСФайлами";
    ...
КонецПроцедуры
```

1. Указать план обмена (РИБ) в типе параметра общей команды `СозданиеНачальногоОбразаСФайлами`. Пример реализации см. в демонстрационной конфигурации в плане обмена `_ДемоОбменВРаспределеннойИнформационной`.

Рассылка отчетов

Подсистема **Рассылка отчетов** позволяет настраивать рассылки вариантов отчетов и отчетов подсистемы **Дополнительные отчеты и обработки**. Рассылки могут выполняться как по расписанию, так и по требованию.

Настройка

После завершения переноса объектов подсистемы необходимо выполнить ее настройку:

- Настроить типы получателей рассылки.
- Расширить состав поддерживаемых форматов файлов.
- Разместить в командном интерфейсе.

Настроить типы получателей рассылки

В качестве получателей рассылки могут выступать различные объекты базы данных с одним общим правилом – для этих объектов должна вестись контактная информация с адресами электронной почты. В процессе настройки рассылки пользователь выбирает тип получателей и вид контактной информации (по умолчанию, это элементы справочников **Пользователи** и **Группы пользователей**).

Для подключения дополнительных типов получателей необходимо:

1. Перечислить типы получателей в свойстве `Тип` определяемого типа `ПолучательРассылки`.
2. В модуле `РассылкаОтчетовПереопределяемый`:

- Опционально. Задать настройки каждого типа через процедуру `ПереопределитьТаблицуТиповПолучателей`: представление, основной вид контактной информации типа E-mail, путь к форме выбора и дополнительный тип, дополняющий элементы основного типа. Если не задать настройки, то будут использованы настройки типа по умолчанию.
- Опционально. Написать код (или изменить запрос) формирования списка получателей в процедуре `ПередФормированиемСпискаПолучателейРассылки`. Эта процедура вызывается непосредственно перед выполнением запроса по получателям. Может потребоваться только в случае использования нестандартной иерархии (например, как в связке справочников `Пользователи` и `ГруппыПользователей`).

Расширить состав поддерживаемых форматов файлов

Если требуется расширить состав форматов, поддерживаемых подсистемой `РассылкаОтчетов`, то необходимо:

1. Добавить новые форматы в перечисление `ФорматыСохраненияОтчетов`.
2. В модуле `РассылкаОтчетовПереопределяемый`:

- Определить параметры формата в процедуре `ПереопределитьПараметрыФорматов`.
- Указать обработчик сохранения в новые форматы в процедуре `ПередСохранениемТабличногоДокументаВФормат`.

Разместить в командном интерфейсе

Для использования подсистемы необходимо разместить командном интерфейсе ответственных за рассылки объект метаданных:

- Справочник `РассылкиОтчетов` – для настройки и интерактивного выполнения рассылок.

В случае если в конфигурации не используется подсистема **Настройки программы**, команду открытия списка справочника `РассылкиОтчетов` следует разместить еще и в рабочем месте администратора приложения.

Подключить нестандартные отчеты

Для участия в рассылках отчеты должны использовать типовой формат хранения и вывода настроек (только средствами СКД) и формирование при помощи метода `СкомпоноватьРезультат` (см. также описание события `ПриКомпоновкеРезультата` в синтакс-помощнике).

Следует найти все отчеты с собственной основной формой и проанализировать, как они формируются и как задаются их настройки.

Если отчет не использует типовой формат хранения и вывода настроек, то следует провести анализ возможности перевода на штатные средства:

- формирование в событии `ПриКомпоновкеРезультата` методом `СкомпоноватьРезультат`,
- настройка при помощи СКД (см. также Подключить форму отчета).

Дополнительно рассылка отчетов позволяет переопределять поведение при выборе отдельных параметров СКД. Для этого в модуле `РассылкаОтчетовКлиентПереопределяемый` определены следующие процедуры:

- `ПриАктивизацииСтрокиНастройки` – позволяет запретить непосредственное редактирование значения настройки – например, если в настройке хранится только представление, а фактически значение хранится в дополнительных свойствах пользовательских настроек СКД.
- `ПриНачалеВыбораНастройки` – позволяет открыть собственную форму для редактирования значения настройки.
- `ПриОчисткеНастройки` – позволяет определить поведение в том случае, если для настройки доступна кнопка очистки.

Если перевод на штатные средства невозможен, то отчет следует добавить в процедуру `ОпределитьИсключаемыеОтчеты` общего модуля `РассылкаОтчетовПереопределяемый`.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Рассылка отчетов» без подсистемы «Управление доступом».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Рассылка отчетов** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`ДобавлениеИзменениеРассылокОтчетов` Настройка и выполнение личных и разрешенных рассылок отчетов. Включает в себя право `Вывод`. Используется совместно с ролью `ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты` (из подсистемы **Работа с почтовыми сообщениями**)
2.	`ЧтениеРассылокОтчетов` Просмотр личных и разрешенных рассылок отчетов.
3.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Смена автора рассылки, настройка и выполнение рассылок отчетов.

Примеры настройки прав доступа пользователей приведены ниже.

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор • Настройка и выполнение рассылок отчетов • Смена ответственных за рассылки отчетов	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Ответственный за рассылки отчетов • Настройка и выполнение рассылок отчетов	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты` (из подсистемы **Работа с почтовыми сообщениями**), • `ДобавлениеИзменениеРассылокОтчетов`.

Использование при разработке конфигурации

Переопределение флажка «ОтчетПустой»

Если для формирования используется собственная логика (в `ПриКомпоновкеРезультата` свойство `СтандартнаяОбработка` устанавливается в `Ложь`), то в процедуре `ПриКомпоновкеРезультата` в дополнительных свойствах пользовательских настроек следует установить признак `ОтчетПустой`:

```
КомпоновщикНастроек.ПользовательскиеНастройки.ДополнительныеСвойства.Вставить("ОтчетПустой", <Истина, если отчет не содержит данных>);
```

Пример № 1. Установка флажка `ОтчетПустой` в отчете, в котором выводятся данные из таблицы значений:

```
ОтчетПустой = ТаблицаЗначений.Количество() = 0;
КомпоновщикНастроек.ПользовательскиеНастройки.ДополнительныеСвойства.Вставить("ОтчетПустой", ОтчетПустой);
```

Пример № 2. Установка флажка `ОтчетПустой` в отчете на СКД, в котором выводятся данные из запросов:

```
ОтчетПустой = ОтчетыСервер.ОтчетПустой(ЭтотОбъект, ПроцессорКД);
КомпоновщикНастроек.ПользовательскиеНастройки.ДополнительныеСвойства.Вставить("ОтчетПустой", ОтчетПустой);
```

Это нужно, чтобы у пользователей работал флажок **Отправлять пустой**, который позволяет отключить отправку пустых отчетов. Для определения, что отчет не пустой, рассылка отчетов использует стандартные средства СКД, которые в некоторых случаях могут не работать или работать недостоверно.

Также этот признак можно использовать и в других случаях, когда требуется переопределить типовую логику или когда рассылка не может достоверно определить, что отчет пустой. Например, если в отчете есть диаграммы, то всегда считается, что отчет не пустой.

Расширение списка параметров темы и текста письма

В ряде случаев стандартных параметров темы и текста письма недостаточно, например, могут потребоваться такие параметры как **Имя**, **Отчество** и т.д. В таком случае их можно определить с помощью общего модуля `РассылкаОтчетовПереопределяемый`:

1. В процедуре `ПриОпределенииПараметровТекстаПисьма` определить новые параметры.
2. В процедуре `ПриПолученииПараметровТекстаПисьма` установить значения для новых параметров.

Пример реализации см. в демонстрационной конфигурации.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между приложениями. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Регламентные задания

Подсистема **Регламентные задания** позволяет редактировать состав и расписание регламентных заданий, просматривать историю выполнения регламентных и фоновых заданий, а также анализировать ошибки при их выполнении.

Настройка

Подсистема не требует специальной настройки.

Размещение в командном интерфейсе

Для использования подсистемы необходимо разместить командном интерфейсе администратора объект метаданных `Обработка.РегламентныеИФоновыеЗадания` – для просмотра фоновых и регламентных заданий, интерактивного выполнения регламентных заданий (настройки обработки регламентных заданий в файловой ИБ).

Настройка зависимостей регламентных заданий

Необходимость выполнение предопределенного регламентного задания можно настраивать в зависимости от различных параметров:

- включения одной или нескольких функциональных опций;
- текущего режима работы - автономное рабочее место, подчиненный узел РИБ, разделенная информационная база;
- по работе с внешними ресурсами.

Подробнее о всех доступных настройках можно посмотреть в комментарии к процедуре `РегламентныеЗаданияПереопределяемый.ПриОпределенииНастроекРегламентныхЗаданий`.

В тех случаях, когда выполнение предопределенного регламентного зависит от описанных выше параметров, необходимо программно управлять признаком `Использование`. Если этого не сделать, регламентное задание будет приводить к запуску сеанса, занимая вычислительные ресурсы сервера **1С:Предприятия**. Для настройки зависимостей регламентного задания на примере функциональных опций необходимо:

1. В состав определяемого типа `МестоХраненияФункциональныхОпций` добавить константы, соответствующие функциональным опциям, используемым для управления регламентными заданиями.
2. Добавить вставку в процедуре `ПриОпределенииНастроекРегламентныхЗаданий` общего модуля `РегламентныеЗаданияПереопределяемый`. Например:

```
Зависимость = Зависимости.Добавить();
Зависимость.РегламентноеЗадание = Метаданные.РегламентныеЗадания.ОбновлениеСтатусовДоставкиSMS;
Зависимость.ФункциональнаяОпция = Метаданные.ФункциональныеОпции.ИспользоватьПочтовыйКлиент;
```

3. Вставить в начало процедуры обработки регламентного задания следующий код:

```
ОбщегоНазначения.ПриНачалеВыполненияРегламентногоЗадания(Метаданные.РегламентныеЗадания.<ИмяРегламентногоЗадания>);
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Регламентные задания** следует использовать роль, указанную ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Добавление и изменение регламентных заданий. Настройка обработки регламентных заданий. Просмотр фоновых заданий.

Использование при разработке конфигурации

При разработке конфигурации никаких дополнительных действий не требуется.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между программами. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Резервное копирование ИБ

Подсистема **Резервное копирование ИБ** позволяет проводить резервное копирование информационной базы из режима **1С:Предприятие** «по требованию» либо в соответствии с настроенным расписанием. Также с помощью данной подсистемы можно проводить восстановление информационной базы из резервной копии.

> **Важно!**
>
> Подсистема недоступна в веб-клиенте.

Настройка

Для использования подсистемы в конфигурации необходимо разместить в командном интерфейсе администратора обработки **Резервное копирование ИБ** и **Настройка резервного копирования ИБ**.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к функциям подсистемы **Резервное копирование ИБ** следует использовать роли, указанные ниже.

№	Роли и их назначение
1.	`АдминистраторСистемы` (из подсистемы **Базовая функциональность**) Выполнение настройки, резервного копирования и восстановления информационной базы.

Использование при разработке конфигурации

При разработке конфигурации никаких дополнительных действий не требуется.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между программами. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Свойства

Подсистема **Свойства** позволяет создавать и редактировать в режиме **1С:Предприятие** дополнительные свойства произвольных объектов конфигурации (не внося изменения в саму конфигурацию). При этом дополнительные свойства можно выводить как в списках, так и в отчетах, как и обычные реквизиты объектов. Объектами со свойствами могут быть любые объекты ссылочного типа.

Дополнительные свойства объектов могут быть трех видов:

- дополнительные реквизиты – это свойства, которые хранятся в самом объекте (в специализированной табличной части). Дополнительные реквизиты являются неотъемлемой частью объекта, вводятся при его редактировании и доступны для редактирования тем же пользователям, которым доступен и сам объект со свойствами. Например, реквизиты `Вес`, `Размеры`, `Цвет` справочника `Товары`;
- метки – это свойства, которые также хранятся в самом объекте (в табличной части дополнительных реквизитов), а представлены на формах в виде разноцветных значков-меток;
- дополнительные сведения – не являются неотъемлемыми свойствами объекта. Они, как правило, доступны для редактирования тем пользователям, которые не имеют доступа к самому объекту. При внедрении подсистемы совместно с подсистемой **Управление доступом** имеется возможность ограничивать доступ пользователей к отдельным дополнительным сведениям. Например, дата ближайшей поставки товара.

Настройка

Перед внедрением подсистемы в конфигурацию необходимо принять решение о составе объектов конфигурации, которые пользователи смогут расширять дополнительными свойствами. В этот список следует включить те объекты, свойства которых с большой вероятностью потребуется определять пользователю. Такими объектами, как правило, являются справочники, которые имеют слишком универсальный характер. Пример – справочник товаров, для которого только при внедрении будет понятна востребованность таких свойств, как вес, размеры, цвет и т. д.

После чего выбрать, какие виды дополнительных свойств необходимы в выбранных объектах:

- дополнительные реквизиты,
- дополнительные сведения,
- метки,
- или все перечисленное.

Затем необходимо принять решение по поводу доступных (для выбора пользователем) типов значений дополнительных свойств и перечислить их в свойстве `Тип` плана видов характеристик `ДополнительныеРеквизитыИСведения`. При работе пользователь сможет создавать дополнительные

свойства таких типов для всех объектов, в которых будет внедрена подсистема **Свойства**. Как правило, в этот список включаются некоторые справочники и перечисления конфигурации. Не рекомендуется включать в состав типов дополнительных свойств такие объекты, как документы, задачи, бизнес-процессы и пр. Также не рекомендуется избыточно расширять состав типов значений дополнительных свойств, т. к. это приведет к усложнению настройки системы пользователем.

Настройка объектов с дополнительными реквизитами

1. Создать табличную часть `ДополнительныеРеквизиты` с реквизитами:

Имя	Тип	Подсказка
`Свойство`	`ПланВидовХарактеристикСсылка.ДополнительныеРеквизитыИСведения`	Дополнительный реквизит
`Значение`	`Характеристика.ДополнительныеРеквизитыИСведения`	Значение дополнительного реквизита
`ТекстоваяСтрока`	`Строка неограниченной длины`	Полный текст строкового дополнительного реквизита

Реквизиту `Значение` в свойстве **Связи параметров выбора** установить связь: `Отбор.Владелец` (`ДополнительныеРеквизиты.Свойство`).

1. В форме объекта рекомендуется создать специальную группу полей или страницу с наименованием `ГруппаДополнительныеРеквизиты` для размещения элементов управления, редактирующих свойства объекта.
2. В обработчике события `ПриСозданииНаСервере` формы объекта необходимо выполнить вызов:

```
// СтандартныеПодсистемы.Свойства
ДополнительныеПараметры = Новый Структура;
ДополнительныеПараметры.Вставить("ИмяЭлементаДляРазмещения", "ГруппаДополнительныеРеквизиты");
УправлениеСвойствами.ПриСозданииНаСервере(ЭтотОбъект, ДополнительныеПараметры);
// Конец СтандартныеПодсистемы.Свойства
```

   где `ГруппаДополнительныеРеквизиты` – имя группы формы, созданной на шаге 2, в которой будут располагаться поля формы, предназначенные для редактирования дополнительных реквизитов. Если этот параметр не указывать, то элементы управления для редактирования свойств будут размещаться в нижней части формы. В некоторых случаях возникает необходимость размещать дополнительные реквизиты в разных местах формы – например, когда в форме с несколькими страницами на каждой из них выводится свой набор свойств. В таком случае в параметр `ИмяЭлементаДляРазмещения` необходимо передать список значений, где значение – ссылка на предопределенный набор свойств, а представление – имя группы формы, в которую нужно выводить его реквизиты. Например:

```
ГруппыДляРазмещения = Новый СписокЗначений;
ГруппыДляРазмещения.Добавить(УправлениеСвойствами.НаборСвойствПоИмени("Справочник_ДемоКонтрагенты_Основное"), Элементы.ГруппаОсновное.Имя);
ГруппыДляРазмещения.Добавить("ВсеОстальные", Элементы.ГруппаПрочее.Имя);
ДополнительныеПараметры = Новый Структура;
ДополнительныеПараметры.Вставить("ИмяЭлементаДляРазмещения", ГруппыДляРазмещения);
УправлениеСвойствами.ПриСозданииНаСервере(ЭтотОбъект, ДополнительныеПараметры);
```

   Значение `ВсеОстальные` в группах для размещения указывает, куда выводить дополнительные реквизиты тех наборов, которые не перечислены в данном списке.
3. При использовании формы для работы с разными объектами следует передать параметр `ПроизвольныйОбъект` со значением `Истина` и в дальнейшем вызывать процедуру `ОбновитьЭлементыДополнительныхРеквизитов` для разных объектов.
4. В модуле формы каждого объекта со свойствами необходимо добавить процедуру:

```
#Область ОбработчикиКомандФормы
// СтандартныеПодсистемы.Свойства
&НаКлиенте
Процедура Подключаемый_СвойстваВыполнитьКоманду(ЭлементИлиКоманда, НавигационнаяСсылка = Неопределено, СтандартнаяОбработка = Неопределено)
    УправлениеСвойствамиКлиент.ВыполнитьКоманду(ЭтотОбъект, ЭлементИлиКоманда, СтандартнаяОбработка);
КонецПроцедуры
// Конец СтандартныеПодсистемы.Свойства
#КонецОбласти
```

5. В обработчике события `ПриОткрытии` добавить код:

```
// СтандартныеПодсистемы.Свойства
    УправлениеСвойствамиКлиент.ПослеЗагрузкиДополнительныхРеквизитов(ЭтотОбъект);
// Конец СтандартныеПодсистемы.Свойства
```

6. В обработчике события `ОбработкаОповещения` добавить код:

```
// СтандартныеПодсистемы.Свойства
Если УправлениеСвойствамиКлиент.ОбрабатыватьОповещения(ЭтотОбъект, ИмяСобытия, Параметр) Тогда
    ОбновитьЭлементыДополнительныхРеквизитов();
    УправлениеСвойствамиКлиент.ПослеЗагрузкиДополнительныхРеквизитов(ЭтотОбъект);
КонецЕсли;
// Конец СтандартныеПодсистемы.Свойства
```

7. Добавить вспомогательные процедуры:

```
#Область СлужебныеПроцедурыИФункции
// СтандартныеПодсистемы.Свойства
&НаСервере
Процедура ОбновитьЭлементыДополнительныхРеквизитов()
    УправлениеСвойствами.ОбновитьЭлементыДополнительныхРеквизитов(ЭтотОбъект);
КонецПроцедуры
&НаКлиенте
Процедура ОбновитьЗависимостиДополнительныхРеквизитов()
    УправлениеСвойствамиКлиент.ОбновитьЗависимостиДополнительныхРеквизитов(ЭтотОбъект);
КонецПроцедуры
&НаКлиенте
Процедура Подключаемый_ПриИзмененииДополнительногоРеквизита(Элемент)
    УправлениеСвойствамиКлиент.ОбновитьЗависимостиДополнительныхРеквизитов(ЭтотОбъект);
КонецПроцедуры
// Конец СтандартныеПодсистемы.Свойства
#КонецОбласти
```

8. В обработчике события `ПриЧтенииНаСервере` добавить код:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
// Конец СтандартныеПодсистемы.Свойства
```

9. В обработчике события `ОбработкаПроверкиЗаполненияНаСервере` добавить код:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ОбработкаПроверкиЗаполнения(ЭтотОбъект, Отказ, ПроверяемыеРеквизиты);
// Конец СтандартныеПодсистемы.Свойства
```

10. В обработчике события `ПередЗаписьюНаСервере` добавить код:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПередЗаписьюНаСервере(ЭтотОбъект, ТекущийОбъект);
// Конец СтандартныеПодсистемы.Свойства
```

Настройка объектов с метками в форме объекта

1. Если в объекте не производилось подключение дополнительных реквизитов, выполнить пункт 1 раздела выше **Настройка объектов с дополнительными реквизитами** по созданию табличной части `ДополнительныеРеквизиты` для хранения меток.
2. Перечислить типы объектов с метками в составе определяемого типа `ВладелецМеток` (это могут быть документы, справочники, планы видов характеристик, планы счетов, планы видов расчета, бизнес-процессы или задачи).
3. В форме объекта рекомендуется создать специальную группу с наименованием `ГруппаМетки` для размещения элементов управления, редактирующих свойства объекта.
4. В обработчике события `ПриСозданииНаСервере` формы объекта необходимо определиться с составом параметров отображения меток, вызвав функцию `УправлениеСвойствами.ПараметрыОтображенияМеток`. Например:

```
ПараметрыОтображенияМеток = УправлениеСвойствами.ПараметрыОтображенияМеток();
ПараметрыОтображенияМеток.ИмяЭлементаДляРазмещенияМеток = "ГруппаМетки";
ПараметрыОтображенияМеток.МаксимумМетокНаФорме = 3;
ПараметрыОтображенияМеток.ВариантОтображенияМеток = Перечисления.ВариантыОтображенияМеток.Картинка;
ДополнительныеПараметры.Вставить("ПараметрыОтображенияМеток", ПараметрыОтображенияМеток);
УправлениеСвойствами.ПриСозданииНаСервере(ЭтотОбъект, ДополнительныеПараметры);
```

где:

- `ИмяЭлементаДляРазмещенияМеток` - имя группы формы, в которой будут размещены метки. Если данный параметр не задан, метки выводятся снизу формы.
- `МаксимумМетокНаФорме` - максимальное количество отображаемых меток на форме. По умолчанию выводятся все без ограничений.
- `ВариантОтображенияМеток` - вариант отображения меток на форме. Миниатюрные цветные картинки для форм с небольшим количеством свободного места или надписи с цветным фоном для быстрого ознакомления с текстом меток. По умолчанию `Перечисления.ВариантыОтображенияМеток.Картинка`.

1. В обработчике события `ОбработкаОповещения` добавить код:

```
// СтандартныеПодсистемы.Свойства
Если УправлениеСвойствамиКлиент.ОбрабатыватьОповещения(ЭтотОбъект, ИмяСобытия, Параметр) Тогда
    ОбновитьЭлементыДополнительныхРеквизитов();
КонецЕсли;
// Конец СтандартныеПодсистемы.Свойства
```

2. Добавить вспомогательную процедуру:

```
#Область СлужебныеПроцедурыИФункции
// СтандартныеПодсистемы.Свойства
&НаСервере
Процедура ОбновитьЭлементыДополнительныхРеквизитов()
    УправлениеСвойствами.ОбновитьЭлементыДополнительныхРеквизитов(ЭтотОбъект);
КонецПроцедуры
// Конец СтандартныеПодсистемы.Свойства
#КонецОбласти
```

См. примеры в демонстрационной конфигурации в справочниках `_ДемоНоменклатура`, `_ДемоКонтрагенты` и `_ДемоПартнеры`.

Настройка объектов с метками в форме списка

Для отображения меток в динамическом списке:

1. В форме списка рекомендуется создать специальную группу колонок с наименованием `ГруппаМетки`.
2. Необходимо определиться с количеством отображаемых колонок меток в форме списка и модифицировать запрос динамического списка. Пример для 3 колонок:

```
...
    0 КАК Метка1,
    0 КАК Метка2,
    0 КАК Метка3
ИЗ
...
```

1. Новые поля необходимо включить в группу колонок.
2. У динамического списка добавить обработчик события `СписокПриПолученииДанныхНаСервере`:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПриПолученииДанныхНаСервере(Настройки, Строки);
// Конец СтандартныеПодсистемы.Свойства
```

В параметр `ИмяВладельца` можно передать имя реквизита, который будет использоваться при получении меток. По умолчанию, отображаются метки ключей строк.

См. примеры в демонстрационной конфигурации в справочниках `_ДемоНоменклатура` и `_ДемоКонтрагенты`.

Для отображения меток при активации строк списка:

1. В форме списка рекомендуется создать специальную группу с наименованием `ГруппаМетки`.
2. В обработчике события `ПриСозданииНаСервере` формы списка необходимо определиться с составом параметров отображения меток, вызвав функцию `УправлениеСвойствами.ПараметрыОтображенияМеток()`. Например:

```
ПараметрыОтображенияМеток = УправлениеСвойствами.ПараметрыОтображенияМеток();
ПараметрыОтображенияМеток.ИмяЭлементаДляРазмещенияМеток = "ГруппаМетки";
ПараметрыОтображенияМеток.ВариантОтображенияМеток = Перечисления.ВариантыОтображенияМеток.Надпись;
ДополнительныеПараметры.Вставить("ПараметрыОтображенияМеток", ПараметрыОтображенияМеток);
ДополнительныеПараметры.Вставить("ПроизвольныйОбъект", Истина);
УправлениеСвойствами.ПриСозданииНаСервере(ЭтотОбъект, ДополнительныеПараметры);
```

где:

- `ИмяЭлементаДляРазмещенияМеток` - имя группы формы, в которой будут размещены метки. Если данный параметр не задан, метки выводятся снизу формы.
- `ВариантОтображенияМеток` - вариант отображения меток на форме. Миниатюрные цветные картинки для форм с небольшим количеством свободного места или надписи с цветным фоном для быстрого ознакомления с текстом меток. По умолчанию `Перечисления.ВариантыОтображенияМеток.Картинка`.

1. Передать параметр `ПроизвольныйОбъект` со значением `Истина` в `ДополнительныеПараметры` процедуры `ПриСозданииНаСервере`.
2. Добавить вспомогательную процедуру:

```
#Область СлужебныеПроцедурыИФункции
&НаСервере
Процедура ЗаполнитьМеткиВФорме()
    Если ЗначениеЗаполнено(ССылкаНаОбъект) Тогда
        УправлениеСвойствами.ЗаполнитьМеткиОбъекта(
            ЭтотОбъект, ССылкаНаОбъект.ПолучитьОбъект(), Истина);
    КонецЕсли;
КонецПроцедуры
#КонецОбласти
```

1. У динамического списка добавить обработчик события `СписокПриАктивизацииСтроки` с вызовом вспомогательной процедуры из пункта 4.

См. пример в демонстрационной конфигурации в справочнике `_ДемоПартнеры`.

Для отображения легенды меток:

1. В форме списка рекомендуется создать специальную группу с наименованием `ГруппаЛегендаМетки`
2. В обработчике события `ПриСозданииНаСервере` формы объекта необходимо определиться с составом параметров отображения легенды меток, вызвав функцию `УправлениеСвойствами.ПараметрыОтображенияМеток()`:

```
ПараметрыОтображенияМеток = УправлениеСвойствами.ПараметрыОтображенияМеток();
ПараметрыОтображенияМеток.ИмяЭлементаДляРазмещенияЛегендыМеток = "ГруппаЛегендаМетки";
ПараметрыОтображенияМеток.ОтборМеток = Истина;
ПараметрыОтображенияМеток.ВидОбъектов = Метаданные.Справочники._ДемоНоменклатура.ПолноеИмя();
```

`ИмяЭлементаДляРазмещенияЛегендыМеток` - имя группы формы, в которой будет размещена легенда меток. Если данный параметр не задан, легенда меток на форме не отображается.

`ОтборМеток` - возможность отбирать объекты в списке по меткам.

`ВидОбъектов` - полное имя объекта метаданных для отображения легенды меток в форме списка.

1. Дополнить структуру новым параметром `ПараметрыОтображенияМеток`:

```
ДополнительныеПараметры.Вставить("ПараметрыОтображенияМеток", ПараметрыОтображенияМеток);
```

1. Добавить вспомогательные процедуры для установки видимости легенды меток:

```
// СтандартныеПодсистемы.Свойства
...
&НаКлиенте
Процедура Подключаемый_УстановитьВидимостьЛегендыМеток(Команда)
    УстановитьВидимостьЛегендыМеток();
КонецПроцедуры

&НаСервере
Процедура УстановитьВидимостьЛегендыМеток()
    УправлениеСвойствами.УстановитьВидимостьЛегендыМеток(ЭтотОбъект);
КонецПроцедуры
...
// Конец СтандартныеПодсистемы.Свойства
```

1. Добавить вспомогательную процедуру для установки отборов по меткам:

```
// СтандартныеПодсистемы.Свойства
...
&НаКлиенте
Процедура Подключаемый_ОбработчикОтбораПоМеткам(Команда)
    УправлениеСвойствамиКлиент.УстановитьОтборПоМетке(ЭтотОбъект, Команда.Имя);
КонецПроцедуры
...
// Конец СтандартныеПодсистемы.Свойства
```

Для установки отборов по меткам важно, чтобы в запросе динамического списка выбиралось поле `Свойство` из таблицы `ДополнительныеРеквизиты`.

См. примеры в демонстрационной конфигурации в справочниках `_ДемоНоменклатура`, `_ДемоКонтрагенты` и `_ДемоПартнеры`.

Настройка отложенной инициализации дополнительных реквизитов

Дополнительные реквизиты, расположенные на отдельной странице (закладке) формы, необходимо загружать отложенно. Это позволит существенно ускорить открытие форм.

> **Важно!**
>
> При отложенной инициализации будет невозможно выполнить перемещение дополнительных реквизитов при помощи команды **Изменить форму** из группы, в которой они расположены.

1. В обработчике события `ПриСозданииНаСервере` формы объекта, в блоке кода:

   ```
   // СтандартныеПодсистемы.Свойства
   ДополнительныеПараметры = Новый Структура;
   ДополнительныеПараметры.Вставить("ИмяЭлементаДляРазмещения", "ГруппаДополнительныеРеквизиты");
   УправлениеСвойствами.ПриСозданииНаСервере(ЭтотОбъект, ДополнительныеПараметры);
   // Конец СтандартныеПодсистемы.Свойства
   ```

   дополнить структуру новым параметром `ОтложеннаяИнициализация`:

   ```
   ДополнительныеПараметры.Вставить("ОтложеннаяИнициализация", Истина);
   ```

2. В обработчике `ПриСменеСтраницы` группы страниц, в которой выводятся дополнительные реквизиты, добавить код:

```
// СтандартныеПодсистемы.Свойства
Если ПараметрыСвойств.Свойство(ТекущаяСтраница.Имя)
        И Не ПараметрыСвойств.ВыполненаОтложеннаяИнициализация Тогда

    СвойстваВыполнитьОтложеннуюИнициализацию();
    УправлениеСвойствамиКлиент.ПослеЗагрузкиДополнительныхРеквизитов(ЭтотОбъект);
КонецЕсли;
// Конец СтандартныеПодсистемы.Свойства
```

3. Добавить вспомогательную процедуру:

```
#Область СлужебныеПроцедурыИФункции
&НаСервере
Процедура СвойстваВыполнитьОтложеннуюИнициализацию()
    УправлениеСвойствами.ЗаполнитьДополнительныеРеквизитыВФорме(ЭтотОбъект);
КонецПроцедуры
#КонецОбласти
```

Настройка объектов с дополнительными сведениями

Перечислить типы объектов с дополнительными сведениями в составе определяемого типа `ВладелецДополнительныхСведений`. Поддерживаемые типы объектов: документы, справочники, планы видов характеристик, планы счетов, планы видов расчета, бизнес-процессы и задачи.

Настройка динамических списков

Для возможности отображения дополнительных реквизитов и сведений в формах списков объектов со свойствами необходимо добавить поле `Ссылка` из запроса динамического списка в таблицу формы, связанную с этим динамическим списком. При этом добавленное поле `Ссылка` следует по умолчанию скрыть от пользователей, сняв флажок `Видимость` в свойстве **Пользовательская видимость**.

Настройка наборов свойств объектов

В большинстве случаев достаточно одного набора свойств для объекта метаданных – например, у всех элементов справочника товаров должны быть такие свойства, как вес, размеры, цвет и т. д. Для этого необходимо:

- В процедуре `ПриПолученииПредопределенныхНаборовСвойств` общего модуля `УправлениеСвойствамиПереопределяемый` описать предопределенный набор свойств с именем по шаблону `Справочник_<ИмяОбъекта>` (если объект – справочник; `Документ_<ИмяОбъекта>`, если объект – документ, и т.д., например, `Справочник_Сотрудники`, `Документ_АвансовыйОтчет`, `БизнесПроцесс_Продажа`), а также указать уникальный идентификатор. Формат уникального идентификатора - Random UUID (Version 4), чтобы получить идентификатор, нужно в режиме **1С:Предприятие** вычислить значение конструктора платформы `Новый УникальныйИдентификатор` или воспользоваться online-генератором (например, https://www.uuidgenerator.net/version4). Например:

  ```
  Набор = Наборы.Строки.Добавить();
  Набор.Имя = "Справочник_ВнешниеПользователи";
  Набор.Идентификатор = Новый УникальныйИдентификатор("<УникальныйИдентификатор>");
  ```

Особые случаи настройки наборов свойств

Перенос наборов свойств из предопределенных элементов справочника НаборыДополнительныхРеквизитовИСведений

Если набор свойств задавался предопределенным элементом справочника `НаборыДополнительныхРеквизитовИСведений`, то необходимо перенести его описание в процедуру `ПриПолученииПредопределенныхНаборовСвойств`, а у предопределенного элемента добавить префикс "Удалить". Например, `УдалитьСправочник_ВнешниеПользователи`.

В таком случае будет создан новый набор свойств и все дополнительные реквизиты и сведения будут перенесены в него. При этом в формах объекта со свойствами могут быть утеряны пользовательские настройки внешнего вида форм (например, размер окна и расположение полей). Для того чтобы не потерять эти настройки, требуется добавить обработчик обновления, в котором вызвать процедуру `УправлениеСвойствами.ВосстановитьНастройкиФормСДополнительнымиРеквизитами`.

Настройка объектов с разными наборами свойств у разных групп объектов

В отдельных случаях может быть необходима более гибкая настройка. Например, в зависимости от вида товара у него могут быть уникальные характеристики: у картриджей для принтеров требуется свойство **Тип принтера**, а у мебели – **Материал**.

При этом подходе набор свойств, применяемый для конкретного экземпляра объекта, определяется по группе объектов, содержащей ссылку на набор свойств. В демонстрационной конфигурации этот вариант применения свойств показан на справочнике `_ДемоНоменклатура`. В каждом элементе справочника применяется тот или иной набор свойств в зависимости от вида номенклатуры (группы объектов). Справочник `_ДемоНоменклатура` имеет реквизит `ВидНоменклатуры`, и в зависимости от его значения элемент справочника получает тот или иной набор дополнительных свойств, ссылка на который извлекается в объекте справочника `_ДемоВидыНоменклатуры`. Для настройки требуется следующее:

1. Создать справочник, значения которого соответствуют разным наборам свойств объекта со свойствами. Например, для справочника `_ДемоНоменклатура` в демонстрационной конфигурации это справочник `_ДемоВидыНоменклатуры`.
2. В этом справочнике добавить реквизит `НаборСвойств` типа `СправочникСсылка.НаборыДополнительныхРеквизитовИСведений`. Включить реквизит в состав функциональной опции (если нет, создать) `ИспользоватьДополнительныеРеквизитыИСведения<Имя конфигурации>`. Если справочник группирует объекты более чем одного типа, допустимо добавить два и более реквизитов с подходящими именами, например: `НаборСвойствОшибок`, `НаборСвойствЗадач`.

3. В обработчике `ПередЗаписью` модуля объекта этого справочника выполнить вызов `УправлениеСвойствами.ПередЗаписьюВидаОбъекта` и передать туда в качестве параметров записываемый объект и имя набора свойств, например:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПередЗаписьюВидаОбъекта(ЭтотОбъект, "Справочник_ДемоНоменклатура");
// Конец СтандартныеПодсистемы.Свойства
```

Если справочник группирует объекты более чем одного типа, следует сделать два или более вызова, указав имена используемых реквизитов, например:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПередЗаписьюВидаОбъекта(ЭтотОбъект, "Справочник_ДемоОшибки","НаборСвойствОшибок");
УправлениеСвойствами.ПередЗаписьюВидаОбъекта(ЭтотОбъект, "Справочник_ДемоЗадачи","НаборСвойствЗадач");
// Конец СтандартныеПодсистемы.Свойства
```

4. В обработчике `ПередУдалением` модуля объекта этого справочника вызывать процедуру `УправлениеСвойствами.ПередУдалениемВидаОбъекта` и передать в качестве параметра записываемый объект, например:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПередУдалениемВидаОбъекта(ЭтотОбъект);
// Конец СтандартныеПодсистемы.Свойства
```

Если справочник группирует объекты более чем одного типа, следует сделать два или более вызова, указав имена используемых реквизитов, например:

```
// СтандартныеПодсистемы.Свойства
УправлениеСвойствами.ПередУдалениемВидаОбъекта(ЭтотОбъект, "НаборСвойствОшибок");
УправлениеСвойствами.ПередУдалениемВидаОбъекта(ЭтотОбъект, "НаборСвойствЗадач");
// Конец СтандартныеПодсистемы.Свойства
```

5. В объекте – владельце свойств создать реквизит, значения которого определяют применение того или иного набора свойств для конкретного экземпляра объекта. Для справочника `_ДемоНоменклатура` это реквизит `ВидНоменклатуры` типа `СправочникСсылка._ДемоВидыНоменклатуры`.
6. В модуле формы объекта реализовать обработчик `ПриИзменении` для элемента формы, редактирующего этот реквизит. Для справочника `_ДемоНоменклатура` это обработчик:

```
&НаКлиенте
Процедура ВидНоменклатурыПриИзменении(Элемент)

    ОбновитьЭлементыДополнительныхРеквизитов();

КонецПроцедуры
```

7. В процедуре `ЗаполнитьНаборыСвойствОбъекта` общего модуля `УправлениеСвойствамиПереопределяемый` вставить условие, заполняющее набор свойств для конкретного объекта. Например, для `_ДемоНоменклатура` это следующее условие:

```
Процедура ЗаполнитьНаборыСвойствОбъекта(Объект, ТипСсылки, НаборыСвойств, СтандартнаяОбработка, КлючНазначения) Экспорт
...
    Если ТипСсылки = Тип("СправочникСсылка._ДемоНоменклатура") Тогда
        ЗаполнитьНаборСвойствПоВидуНоменклатуры(
            Объект, ТипСсылки, НаборыСвойств);
    КонецЕсли;
...
КонецПроцедуры
Процедура ЗаполнитьНаборСвойствПоВидуНоменклатуры(Номенклатура, ТипСсылки, НаборыСвойств)

    Если ТипЗнч(Номенклатура) = ТипСсылки Тогда
        Номенклатура = ОбщегоНазначения.ЗначенияРеквизитовОбъекта(
            Номенклатура, "ЭтоГруппа, ВидНоменклатуры");
    КонецЕсли;

    Если Номенклатура.ЭтоГруппа = Ложь Тогда
        Строка = НаборыСвойств.Добавить();
        Строка.Набор = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(
            Номенклатура.ВидНоменклатуры, "НаборСвойств");
    КонецЕсли;

КонецПроцедуры
```

Если справочник группирует объекты более чем одного типа, следует вписать в условие несколько типов, например:

```
Если ТипСсылки = Тип("СправочникСсылка._ДемоОшибки")
       ИЛИ ТипСсылки = Тип("СправочникСсылка._ДемоЗадачи") Тогда
    ЗаполнитьНаборСвойствПоПроекту(Объект, ТипСсылки, НаборыСвойств);
КонецЕсли;
```

8. В процедуре `ПриПолученииПредопределенныхНаборовСвойств` общего модуля `УправлениеСвойствамиПереопределяемый` описать предопределенную группу с именем `Справочник_<ИмяОбъекта>`, если объект – справочник; `Документ_<ИмяОбъекта>`, если объект – документ, и т. д. Например, для

справочника `Номенклатура` создана предопределенная группа `Справочник_Номенклатура`:

```
Набор = Наборы.Строки.Добавить();
Набор.Имя = "Справочник_Номенклатура";
Набор.ЭтоГруппа = Истина;
Набор.Идентификатор = Новый УникальныйИдентификатор("c7cd91d8-6f8a-4d10-82bf-c6fba8475a98");
```

Настройка объектов с несколькими объединяющимися наборами свойств

Наборы свойств при этом должны объединяться в зависимости от тех или иных реквизитов объекта. Например, в справочнике **Партнеры** могут быть введены клиенты, поставщики, конкуренты и партнеры, с которыми связывают прочие отношения. При этом один и тот же партнер может одновременно быть и тем, и другим, и третьим, а его дополнительные свойства должны динамически выводиться в зависимости от того, каких он типов.

В демонстрационной конфигурации этот вариант показан на справочнике `_ДемоПартнеры`. В нем введены реквизиты `Клиент`, `Поставщик`, `Конкурент`, `ПрочиеОтношения` типа `Булево`, которые и определяют применение одного или сразу нескольких наборов свойств.

При этом подходе справочник имеет один предопределенный набор свойств для всех экземпляров объектов, а также несколько предопределенных наборов свойств, которые могут применяться к экземплярам данного объекта по тем или иным условиям.

Последовательность настройки такого объекта:

1. Принять решение по поводу наборов свойств объекта. Например, для справочника `_ДемоПартнеры` демонстрационной конфигурации это четыре набора свойств: реквизиты `Клиент`, `Поставщик`, `Конкурент`, `ПрочиеОтношения`.
2. Выявить реквизиты объекта, значения которых определяют применение того или иного набора свойств для конкретного экземпляра объекта. Например, для справочника `_ДемоПартнеры` демонстрационной конфигурации это реквизиты `Клиент`, `Поставщик`, `Конкурент`, `ПрочиеОтношения`.
3. В процедуре `ПриПолученииПредопределенныхНаборовСвойств` общего модуля `УправлениеСвойствамиПереопределяемый` описать предопределенную группу с именем `Справочник_<ИмяОбъекта>`, если объект – справочник; `Документ_<ИмяОбъекта>`, если объект – документ, и т. д. Например, `Справочник__ДемоПартнеры`. В этой предопределенной группе для каждого из наборов свойств описать предопределенный элемент с именами `Справочник_<ИмяОбъекта>_<ИмяНабораСвойств>`, например `Справочник_Партнеры_Клиенты`.

```
Набор = Наборы.Строки.Добавить();
Набор.Имя = "Справочник__ДемоПартнеры";
Набор.ЭтоГруппа = Истина;
Набор.Идентификатор = Новый УникальныйИдентификатор("2c8d6c08-1d35-43ce-a690-32ccf53b03f2");
ДочернийНабор = Набор.Строки.Добавить();
ДочернийНабор.Имя = "Справочник_Партнеры_Клиенты";
ДочернийНабор.Идентификатор = Новый УникальныйИдентификатор("277e1f06-e4f6-4c23-8d31-0d4347e207a2");
```

4. Наименование дочерних наборов свойств (`Справочник_Партнеры_Клиенты` и т.д.) необходимо указывать в процедуре `ПриПолученииНаименованийНаборовСвойств` общего модуля `УправлениеСвойствамиПереопределяемый` по примеру:

```
Наименования["Справочник_Партнеры_Клиенты"]    = НСтр("ru='Клиент'; en='Client';", КодЯзыка);
Наименования["Справочник_Партнеры_Конкуренты"] = НСтр("ru='Конкурент'; en='Competitor';", КодЯзыка);
Наименования["Справочник_Партнеры_Общие"]      = НСтр("ru='Общие'; en='General';", КодЯзыка);
Наименования["Справочник_Партнеры_Поставщики"] = НСтр("ru='Поставщик'; en='Provider';", КодЯзыка);
Наименования["Справочник_Партнеры_Прочие"]     = НСтр("ru='Прочее'; en='Other';", КодЯзыка);
```

5. В этой предопределенной группе также создать предопределенный элемент с именем `Справочник_<ИмяОбъекта>_Общие` и наименованием `Общие`, который будет содержать общие свойства для всех объектов.
6. В общем модуле `УправлениеСвойствамиПереопределяемый` в процедуре `ЗаполнитьНаборыСвойствОбъекта` следует вписать условие для данного типа объектов. Для справочника `_ДемоПартнеры` это код вида:

```
Процедура ЗаполнитьНаборыСвойствОбъекта(Объект, ТипСсылки, НаборыСвойств, СтандартнаяОбработка) Экспорт
...
    Если ТипСсылки = Тип("СправочникСсылка._ДемоПартнеры") Тогда
        ЗаполнитьНаборыСвойствПартнера(Объект, ТипСсылки, НаборыСвойств);
    КонецЕсли;
...
КонецПроцедуры
Процедура ЗаполнитьНаборыСвойствПартнера(Партнер, ТипСсылки, НаборыСвойств)

    Если ТипЗнч(Партнер) = ТипСсылки Тогда
        Объект = ОбщегоНазначения.ЗначенияРеквизитовОбъекта(
            Партнер, "Клиент, Конкурент, Поставщик, ПрочиеОтношения, ЭтоГруппа");
    Иначе
        Объект = Партнер;
    КонецЕсли;

    Если Объект.ЭтоГруппа = Ложь Тогда

        Строка = НаборыСвойств.Добавить();
        Строка.Набор = УправлениеСвойствами.НаборСвойствПоИмени("Справочник_Партнеры_Общие");
        Строка.Отображение = ОтображениеОбычнойГруппы.Линия;
        Строка.ОтображатьЗаголовок = Истина;
        Строка.Заголовок = НСтр("ru = 'Для всех партнеров'");

        Если Объект.Клиент Тогда
            Строка = НаборыСвойств.Добавить();
            Строка.Набор = УправлениеСвойствами.НаборСвойствПоИмени("Справочник_Партнеры_Клиенты");
            Строка.Отображение = ОтображениеОбычнойГруппы.Линия;
            Строка.ОтображатьЗаголовок = Истина;
            Строка.Заголовок = НСтр("ru = 'Для клиентов'");
        КонецЕсли;
…
КонецПроцедуры
```

В приведенном примере проверяются значения реквизитов справочника `Клиент`, `Поставщик`, `Конкурент`, `ПрочиеОтношения`, и наборы свойств добавляются в таблицу `НаборыСвойств`. В поле `Набор` устанавливается ссылка на набор свойств, а в остальные поля могут быть установлены значения оформления набора в форме.

7. В форме объекта необходимо реализовать обработчики `ПриИзменении` для тех реквизитов, которые определят состав наборов свойств, применяемых для данного экземпляра объекта. Например, для формы справочника `_ДемоПартнеры` это обработчики вида:

```
&НаКлиенте
Процедура КлиентПриИзменении(Элемент)

    // СтандартныеПодсистемы.Свойства
    ОбновитьЭлементыДополнительныхРеквизитов();
    // Конец СтандартныеПодсистемы.Свойства

КонецПроцедуры
```

Настройка дополнительных характеристик объектов метаданных

Для каждого объекта конфигурации со свойствами необходимо открыть и заполнить диалог редактирования дополнительных характеристик.

Если для объекта настроены дополнительные реквизиты:

Виды характеристик	Поле ключа	Поле отбора видов	Зн
`Справочник.НаборыДополнительныхРеквизитовИСведений.ТабличнаяЧасть.ДополнительныеРеквизиты`	`Свойство`	`ИмяПредопределенногоНабора`	Имя предоп описанного `ПриПолучени` модуля `УправлениеС`

Значения характеристик	Поле объекта	Поле вида	Поле значения
`Справочник.<Имя объекта со свойствами>.ТабличнаяЧасть.ДополнительныеРеквизиты`	`Ссылка`	`Свойство`	`Значение`

Если для объекта настроены дополнительные сведения:

Виды характеристик	Поле ключа	Поле отбора видов	Зна
`Справочник.НаборыДополнительныхРеквизитовИСведений.ТабличнаяЧасть.ДополнительныеСведения`	`Свойство`	`ИмяПредопределенногоНабора`	Имя предопр описанного в `ПриПолучении` модуля `УправлениеСв`

Значения характеристик	Поле объекта	Поле вида	Поле значения
`РегистрСведений.ДополнительныеСведения`	`Объект`	`Свойство`	`Значение`

где **Предопределенный элемент (группа)** справочника `НаборыДополнительныхРеквизитовИСведений` – предопределенный элемент (группа), созданный на предыдущем этапе.

Отключение неиспользуемых наборов свойств

В случае если объект, использующий набор свойств, отключен по функциональной опции, имеется возможность отключить неиспользуемый набор свойств. Для этого можно воспользоваться свойством `Используется` набора свойств. При установке значения свойства `Используется` в `Ложь` набор свойств перестает отображаться в форме списка, также свойства перестают отображаться в форме объекта-владельца. Для отключения неиспользуемого набора свойств необходимо:

- В процедуре `ПриПолученииПредопределенныхНаборовСвойств` общего модуля `УправлениеСвойствамиПереопределяемый` для нужного набора свойств установить значение свойства `Используется`:

```
Набор = Наборы.Строки.Добавить();
Набор.Имя = "Справочник_ВнешниеПользователи";
Набор.Идентификатор = Новый УникальныйИдентификатор("d9c30d48-a72a-498a-9faa-c078bf652776");
Набор.Используется  = ПолучитьФункциональнуюОпцию("ИспользоватьВнешнихПользователей");
```

- Добавить код по переключению использования набора свойств в событие при изменении значения константы.

```
ПараметрыНабора = УправлениеСвойствами.СтруктураПараметровНабораСвойств();
ПараметрыНабора.Используется = ПолучитьФункциональнуюОпцию("ИспользоватьВнешнихПользователей");
УправлениеСвойствами.УстановитьПараметрыНабораСвойств("Справочник_ВнешниеПользователи", ПараметрыНабора);
```

Размещение в командном интерфейсе

Для использования подсистемы необходимо разместить в командном интерфейсе объект метаданных:

- Справочник `НаборыДополнительныхРеквизитовИСведений` – для работы по просмотру и редактированию состава свойств (редактором состава свойств и администратором).

В случае если в конфигурации не используется подсистема **Настройки программы**, в рабочем месте администратора приложения необходимо разместить константы:

- `ИспользоватьДополнительныеРеквизитыИСведения` – для включения и выключения подсистемы;

См. пример в форме `ОбщиеНастройки` обработки `ПанельАдминистрированияБСП`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы следует использовать роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` Просмотр наборов свойств, дополнительных свойств и значений свойств объектов.
2.	`ЧтениеДополнительныхСведений` Просмотр разрешенных дополнительных сведений. При совместном использовании с подсистемой **Управление доступом** имеется возможность ограничивать доступ пользователей к отдельным дополнительным сведениям. См. раздел Особые случаи внедрения подсистемы.
3.	`ИзменениеДополнительныхСведений` Добавление и изменение разрешенных дополнительных сведений. При совместном использовании с подсистемой **Управление доступом** имеется возможность ограничивать доступ пользователей к отдельным дополнительным сведениям. См. раздел Особые случаи внедрения подсистемы.
4.	`ДобавлениеИзменениеДополнительныхРеквизитовИСведений` Добавление и изменение свойств, наборов свойств, значений свойств объектов.
5.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение свойств. Удаление помеченных на удаление объектов подсистемы.

Для просмотра и редактирования дополнительных реквизитов и меток конкретных объектов требуются роли для просмотра и редактирования самих объектов-владельцев дополнительных реквизитов и меток.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за настройку состава дополнительных свойств	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеДополнительныхРеквизитовИСведений`, • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**)
2.	Просмотр значений дополнительных сведений объектов информационной базы	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЧтениеДополнительныхСведений`, • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**)
3.	Редактирование значений дополнительных сведений объектов информационной базы	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ИзменениеДополнительныхСведений`, • `ЗапускТонкогоКлиента` (из подсистемы **Базовая функциональность**)

Программная работа со свойствами

У каждого свойства есть реквизит **Имя**, которое необходимо для обращения к нему из кода с помощью функций `ЗначенияСвойств`, `ЗначениеСвойства`, `ПредставлениеЗначенияСвойства` и `ПредставленияЗначенийСвойств` общего модуля `УправлениеСвойствами`. Его значение уникально, задается автоматически и при необходимости может быть изменено разработчиком в карточке свойства.

Кроме того, можно заполнять параметры печатной формы, имена которых соответствуют именам дополнительных реквизитов, следующим образом:

```
Макет = УправлениеПечатью.МакетПечатнойФормы("Документ._ДемоСчетНаОплатуПокупателю.ПФ_MXL_ГарантийноеПисьмо", КодЯзыка);
ПредставленияЗначенийСвойств = УправлениеСвойствами.ПредставленияЗначенийСвойств(МассивОбъектов, КодЯзыка);

Для Каждого Документ Из ДанныеПечати Цикл
    ...
    ОбластьМакета = Макет.ПолучитьОбласть("ТекстПисьма");
    ...
    ОбластьМакета.Параметры.Заполнить(ПредставленияЗначенийСвойств[Документ.Ссылка]);

    ТабличныйДокумент.Вывести(ОбластьМакета);

    ...
КонецЦикла;
```

Полный список доступных и функций см. в главе 4 «Программный интерфейс».

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Свойства» без подсистемы «Взаимодействия».
- Внедрение подсистемы «Свойства» без подсистемы «Работа с файлами».
- Внедрение подсистем «Свойства» и «Управление доступом».

Использование при разработке конфигурации

При создании новых прикладных объектов метаданных нужно повторно выполнять соответствующую часть процедуры настройки, описанной выше.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, содержащие данные.

Склонение представлений объектов

Подсистема **Склонение представлений объектов** предназначена для автоматического склонения представлений объектов с возможностью ручной корректировки пользователем.

Настройка

Принять решение по поводу объектов метаданных конфигурации ссылочного типа, представления которых следует склонять.

Если в конфигурации не используется подсистема **Настройки программы**, тогда на рабочем месте администратора приложения (администратора сервиса, если конфигурация работает в модели сервиса) необходимо разместить константу:

- `ИспользоватьСервисСклоненияMorpher`,

Реквизит `ТокенДоступаКСервисуMorpher` нужно разместить на форме по правилам безопасного хранения паролей (см. документацию к подсистеме **Базовая функциональность**).

См. пример в форме `ОбщиеНастройки` обработки `ПанельАдминистрированияБСП`.

Для каждого объекта метаданных, который определен для встраивания подсистемы склонения объектов, необходимо:

- Все склоняемые объекты перечислить в свойстве `Тип` определяемого типа `ОбъектСклонения`.
- Во всех формах элементов, для которых встраивается склонение, в обработчике `ПриСозданииНаСервере` добавить фрагмент кода:

  ```
  // СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  СклонениеПредставленийОбъектов.ПриСозданииНаСервере(ЭтотОбъект, Представление);
  // Конец СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  ```

  В параметре `Представление` указывается реквизит, который должен склоняться.
- Во всех формах элементов, для которых встраивается склонение, в обработчике `ПриЗаписиНаСервере` добавить фрагмент кода:

  ```
  // СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  СклонениеПредставленийОбъектов.ПриЗаписиФормыОбъектаСклонения(ЭтотОбъект, Представление, ТекущийОбъект.Ссылка, ПараметрыСклонения);
  // Конец СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  ```

  В параметре `Представление` указывается реквизит, который должен склоняться. В параметре `ПараметрыСклонения` необязательно передается структура дополнительных параметров склонения. Для конструирования структуры используется функция `ПараметрыСклонения` общего модуля `СклонениеПредставленийОбъектовКлиентСервер`.
- Во всех формах элементов, для которых встраивается склонение, рядом с реквизитом, который необходимо просклонять, добавить команду в виде гиперссылки с названием **Склонения**. Обработчик команды:

  ```
  // СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  &НаКлиенте
  Процедура Склонения(Команда)
      СклонениеПредставленийОбъектовКлиент.ПоказатьСклонение(ЭтотОбъект, Представление, ПараметрыСклонения);
  КонецПроцедуры
  // Конец СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  ```

  В параметре `Представление` указывается реквизит, который должен склоняться. В параметре `ПараметрыСклонения` необязательно передается структура дополнительных параметров склонения. Для конструирования структуры используется функция `ПараметрыСклонения` общего модуля `СклонениеПредставленийОбъектовКлиентСервер`.
- Во всех формах элементов, для которых встраивается склонение, в обработчике `ПриИзменении` реквизита, который должен склоняться, добавить фрагмент кода:

  ```
  // СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  СклонениеПредставленийОбъектовКлиент.ПросклонятьПредставление(ЭтотОбъект, Представление, ПараметрыСклонения);
  // Конец СтандартныеПодсистемы.СклонениеПредставленийОбъектов
  ```

  В параметре `Представление` указывается реквизит, который должен склоняться. В параметре `ПараметрыСклонения` необязательно передается структура дополнительных параметров склонения. Для конструирования структуры используется функция `ПараметрыСклонения` общего модуля `СклонениеПредставленийОбъектовКлиентСервер`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Склонение представлений объектов** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Включение и отключение использования веб-сервиса склонения **Morpher.ru**, настройка доступа к сервису. Полный доступ к регистру сведений **Склонения представлений объектов**.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Склонения представлений объектов хранятся в регистре сведений **Склонения представлений объектов**.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, содержащие данные.

Структура подчиненности

Подсистема **Структура подчиненности** предоставляет отчет о взаимосвязях родительских и подчиненных документов. Например, просматривать цепочки связанных документов для заказов, договоров и т.п.

Настройка

Принять решение по поводу списка документов, справочников и планов видов характеристик (далее - документы), для которых требуется выводить отчет **Связанные документы**. Например, заказы, договоры и т.п.

- Если в конфигурацию внедрена подсистема **Подключаемые команды**, то нужно предварительно встроить эту подсистему в интересующий документ.
- Если в конфигурацию внедрена подсистема **Подключаемые команды** и она уже встроена в интересующий документ, то дополнительных действий не требуется: команда для открытия отчета **Связанные документы** автоматически размещается в подменю **Отчеты**.
- Если в конфигурации отсутствует подсистема **Подключаемые команды**, то задать список типов выбранных документов в типах параметра общей команды `СвязанныеДокументы`.

Для настройки связи между родительскими и подчиненными документами, справочниками и ПВХ, выводимыми в отчет **Связанные документы**, необходимо настроить критерий отбора `СвязанныеДокументы`. В свойстве `Тип` требуется указать возможные типы родительских документов, а в свойстве `Состав` – реквизиты подчиненных документов, справочников и планов видов характеристик, в которых будет происходить поиск родительских документов. Например, указать, что в реквизите `Договор` заказа покупателя находится ссылка на родительский договор.

Настройка переопределяемых модулей

При необходимости вписать реализацию в функции переопределяемого модуля подсистемы `СтруктураПодчиненностиПереопределяемый`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы следует использовать следующую роль:

№	Роли и их назначение
1.	`ПросмотрСвязанныеДокументы` Просмотр документов, связанных с интересующим объектом приложения.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Текущие дела

Подсистема **Текущие дела** позволяет выводить список дел пользователя на рабочем столе (новые письма, задачи, заявки, несогласованные заказы и т. п.). Список дел определяется разработчиком конфигурации при внедрении подсистемы и может включать в себя важные и обычные дела, однострочные и многострочные, с количественными показателями и без. Список дел сгруппирован по разделам командного интерфейса, предусмотрена возможность настраивать видимость и порядок отображения, автоматическое обновление через заданный интервал.

Настройка

Разместить основную форму обработки `ТекущиеДела` в правой колонке рабочей области начальной страницы.

Принять решение по поводу состава дел, выводимых на рабочем столе. Например, это могут быть уведомления о новых письмах, задачах, заявках, несогласованных заказах и других делах, с которыми сталкиваются пользователи по своим должностным обязанностям. С каждым делом необходимо связать форму списка или рабочего места, которое должно открываться по нажатию на гиперссылку дела.

Рекомендуется группировать дела по разделам командного интерфейса, в которых размещены команды открытия рабочих мест по разбору этих дел. Например, если работа с письмами и задачами предусмотрена в разделе **Органайзер**, то соответствующие им дела также следует размещать в этом разделе панели **Текущие дела**.

При этом с помощью процедуры `ПриОпределенииПорядкаРазделовКомандногоИнтерфейса` общего модуля `ТекущиеДелаПереопределяемый` можно указать порядок следования разделов панели, который соответствует порядку в интерфейсе приложения.

Добавление нового дела

Для добавления нового дела на панель **Текущие дела** необходимо в процедуре `ПриОпределенииОбработчиковТекущихДел` общего модуля `ТекущиеДелаПереопределяемый` объявить общий модуль или модуль менеджера объекта, который будет предоставлять сведения об этом деле. Например:

```
Процедура ПриОпределенииОбработчиковТекущихДел(Обработчики) Экспорт

    Обработчики.Добавить(Документы._ДемоЗаказПокупателя);

КонецПроцедуры
```

После этого необходимо в указанном модуле определить экспортную процедуру `ПриЗаполненииСпискаТекущихДел` по шаблону:

```
// Заполняет список текущих дел пользователя.
//
// Параметры:
//  ТаблицаЗначений - определяет параметры дела:
//    * Идентификатор  - Строка - внутренний идентификатор дела, используемый подсистемой.
//    * ЕстьДела       - Булево - если Истина, дело выводится в списке текущих дел пользователя.
//    * Важное         - Булево - если Истина, дело будет выделено красным цветом.
//    * ВыводитьВОповещениях - Булево - если Истина, уведомление о деле будет дублироваться всплывающим
//                             оповещением и отображением в центре оповещений.
//    * СкрыватьВНастройках - Булево - если Истина, то дело будет скрыто в форме настроек текущих дел.
//                            Можно применять для дел, которые не предполагают многократного
//                            использования, т.е. после выполнения они для данной информационной базы
//                            больше отображаться не будут.
//    * Представление  - Строка - представление дела, выводимое пользователю.
//    * Количество     - Число  - количественный показатель дела, выводится в строке заголовка дела.
//    * Форма          - Строка - полный путь к форме, которую необходимо открыть при нажатии на гиперссылку
//                                дела на панели "Текущие дела".
//    * ПараметрыФормы - Структура - параметры, с которыми нужно открывать форму показателя.
//    * Владелец       - Строка
//                     - ОбъектМетаданных - строковый идентификатор дела, которое будет владельцем для текущего
//                       или объект метаданных подсистема.
//    * Подсказка      - Строка - текст подсказки.
//    * ОбъектВладелецДел - Строка - полное имя объекта метаданных, в котором расположен обработчик заполнения дел.
//
Процедура ПриЗаполненииСпискаТекущихДел(ТекущиеДела) Экспорт
КонецПроцедуры
```

И вписать в нее код по добавлению дел в таблицу `ТекущиеДела`.

> **Важно!**
>
> При добавлении дела необходимо проверять право `Редактирование` текущего пользователя на объект метаданных, предоставляющий это дело, а также значения функциональных опций (если они предусмотрены).

Пример заполнения дела по документам `_ДемоЗаказПокупателя`:

```
Процедура ПриЗаполненииСпискаТекущихДел(ТекущиеДела) Экспорт

    Если Не ПравоДоступа("Редактирование", Метаданные.Документы._ДемоЗаказПокупателя) Тогда
        Возврат;
    КонецЕсли;

    КоличествоЗаказовПокупателя = КоличествоЗаказовПокупателя();

    СписокОтбора = Новый СписокЗначений;
    СписокОтбора.Добавить(Перечисления._ДемоСтатусыЗаказовПокупателей.НеСогласован);
    СписокОтбора.Добавить(Перечисления._ДемоСтатусыЗаказовПокупателей.Согласован);

    ИдентификаторЗаказыПокупателя = "ЗаказыПокупателя";
    Дело = ТекущиеДела.Добавить();
    Дело.Идентификатор  = ИдентификаторЗаказыПокупателя;
    Дело.ЕстьДела       = КоличествоЗаказовПокупателя.Всего > 0;
    Дело.Представление  = НСтр("ru = 'Заказы покупателя'");
    Дело.Количество     = КоличествоЗаказовПокупателя.Всего;
    Дело.Форма          = "Документ._ДемоЗаказПокупателя.Форма.ФормаСписка";
    Дело.ПараметрыФормы = Новый Структура("Отбор", Новый Структура("СтатусЗаказа", СписокОтбора));
    Дело.Владелец       = Метаданные.Подсистемы._ДемоОрганайзер;

    Дело = ТекущиеДела.Добавить();
    Дело.Идентификатор  = "ЗаказыПокупателяНеСогласовано";
    Дело.ЕстьДела       = КоличествоЗаказовПокупателя.НеСогласовано > 0;
    Дело.Важное         = Истина;
    Дело.Представление  = НСтр("ru = 'Не согласовано'");
    Дело.Количество     = КоличествоЗаказовПокупателя.НеСогласовано;
    Дело.Владелец       = ИдентификаторЗаказыПокупателя;

    Дело = ТекущиеДела.Добавить();
    Дело.Идентификатор  = "ЗаказыПокупателяСогласовано";
    Дело.ЕстьДела       = КоличествоЗаказовПокупателя.Согласовано > 0;
    Дело.Представление  = НСтр("ru = 'Согласовано'");
    Дело.Количество     = КоличествоЗаказовПокупателя.Согласовано;
    Дело.Владелец       = ИдентификаторЗаказыПокупателя;

КонецПроцедуры
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к функциям подсистемы **Текущие дела** нужно использовать следующую роль:

№	Роли и их назначение
1.	`ИспользованиеТекущихДел` Просмотр и настройка списка текущих дел.

Использование при разработке конфигурации

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Удаление помеченных объектов

Подсистема **Удаление помеченных объектов** предназначена для безвозвратного удаления объектов информационной базы.

Удаление помеченных объектов можно выполнять по расписанию.

Также подсистема позволяет вывести в формы списков справочников, планов видов характеристик и т.д. элементы интерфейса для управления видимостью помеченных на удаления объектов.

Настройка

Настройка скрытия помеченных объектов в списках

Необходимо принять решение по перечню метаданных и составу форм, в списках которых требуется скрывать элементы, помеченные на удаление и в которых будут выведены команды по управлению видимостью помеченных на удаления объектов.

Рекомендуется выполнить данную настройку во всех формах списков справочников, документов, планов видов характеристик, планов счетов и планов видов расчетов, а также в обработках со списками элементов, где пользователям в обычных сценариях работы не требуется видеть элементы, помеченные на удаление. Однако если такая необходимость возникает, то показать помеченные на удаление возможно с помощью соответствующей команды в меню **Еще**.

Команды для управления видимостью помеченных на удаления объектов могут быть добавлены на формы с использованием подсистемы `ПодключаемыеКоманды` (рекомендуемый способ) или с использованием программного интерфейса подсистемы.

Если подсистема `ПодключаемыеКоманды` используется, то необходимо перечислить выбранные объекты в процедуре `ПриОпределенииОбъектовСКомандойПоказатьПомеченные` общего модуля `УдалениеПомеченныхОбъектовПереопределяемый`.

В выбранных формах в обработчике `ПриСозданииНаСервере` вызвать процедуру `ПриСозданииНаСервере` общего модуля `УдалениеПомеченныхОбъектов`:

- для формы с одним динамическим списком:

```
УдалениеПомеченныхОбъектов.ПриСозданииНаСервере(ЭтотОбъект, Элементы.Список);
```

- для формы с несколькими списками:

```
НастройкиОтображенияПомеченных = УдалениеПомеченных.НастройкиОтображенияПомеченныхОбъектов();
Настройка = НастройкиОтображенияПомеченных.Добавить();
Настройка.ИмяЭлементаФормы = Элементы.Список1.Имя;
Настройка = НастройкиОтображенияПомеченных.Добавить();
Настройка. ИмяЭлементаФормы = Элементы.Список2.Имя;
УдалениеПомеченных.ПриСозданииНаСервере(ЭтотОбъект, НастройкиОтображенияПомеченных);
```

Если в объекте не используется подсистема `ПодключаемыеКоманды`, есть возможность вывести команды **Показать/скрыть помеченные на удаление** и **Перейти к помеченным на удаление** на форму:

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    // Удаление помеченных
    УдалениеПомеченныхОбъектов.ПриСозданииНаСервере(ЭтотОбъект, НастройкиОтображенияПомеченных);
    УдалениеПомеченныхОбъектов.УстановитьПометкуКомандыПоказатьПомеченные(ЭтотОбъект, Элементы.Список, Элементы.ПоказатьПомеченныеНаУдаление);
    // Удаление помеченных Конец
КонецПроцедуры
&НаКлиенте
Процедура ПерейтиКПомеченнымНаУдаление(Команда)
    УдалениеПомеченныхОбъектовКлиент.ПерейтиКПомеченнымНаУдаление(ЭтотОбъект, Элементы.Список);
КонецПроцедуры
&НаКлиенте
Процедура ПоказатьПомеченныеНаУдаление(Команда)
    КнопкаФормы = Элементы.ПоказатьПомеченныеНаУдаление;
    УдалениеПомеченныхОбъектовКлиент.ПоказатьПомеченныеНаУдаление(ЭтотОбъект, Элементы.Список, КнопкаФормы);
КонецПроцедуры
```

Размещение настроек подсистемы в собственных формах

В случае, если в конфигурации не используется подсистема **Настройки программы**, в рабочем месте администратора приложения необходимо разместить обработку **Удаление помеченных объектов**.

См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

Для размещения настроек подсистемы в произвольной форме следует:

- создать реквизит формы `АвтоматическиУдалятьПомеченныеОбъекты` типа `Булево` для управления флажком настроек, который следует разместить в элементах формы;
- в обработчике события формы `ПриСозданииНаСервере` вызвать процедуру `РежимУдалятьПоРасписанию` общего модуля `УдалениеПомеченныхОбъектов`:

```
Если ОбщегоНазначения.ПодсистемаСуществует("СтандартныеПодсистемы.УдалениеПомеченныхОбъектов")
        И Пользователи.ЭтоПолноправныйПользователь(,Истина) Тогда

    МодульУдалениеПомеченныхОбъектов = ОбщегоНазначения.ОбщийМодуль("УдалениеПомеченныхОбъектов");
    АвтоматическиУдалятьПомеченныеОбъекты = МодульУдалениеПомеченныхОбъектов.РежимУдалятьПоРасписанию().Использование;
КонецЕсли;
```

- при изменении реквизита формы `АвтоматическиУдалятьПомеченныеОбъекты` вызвать процедуру `ПриИзмененииФлажкаУдалятьПоРасписанию` общего модуля `УдалениеПомеченныхОбъектовКлиент`.

Очистка мест использования при удалении помеченных

При удалении помеченных объектов производится контроль ссылочной целостности базы путем поиска ссылок на удаляемый объект. Если на удаляемый объект ссылаются другие объекты приложения, то объект не удаляется, а пользователю выводятся результаты поиска ссылок на удаляемый объект.

При этом удаление не блокируют ссылки из измерений регистров с признаком `Ведущее` (для измерений регистров сведений с признаком `Ведущее` специальная обработка не требуется, поскольку при удалении ссылки все записи по этому измерению удаляются автоматически) и ссылки из объектов метаданных, указанных как подчиненные. Подробнее про подчиненные объекты см. в разделе Поиск и удаление дублей.

В других случаях, когда ссылки между объектами информационной базы не являются значимыми для таких операций, как удаление объекта или поиск ссылок на объект, необходимо:

- настроить список исключений поиска ссылок в функции `ПриДобавленииИсключенийПоискаСсылок` в переопределяемом модуле `ОбщегоНазначенияПереопределяемый`. Подробнее про исключения поиска ссылок см. в разделе Базовая функциональность;

- обеспечить очистку ссылок удаляемого объекта из данных объектов-исключений в обработчике `ПередУдалением` удаляемого объекта (или с помощью подписки на одноименное событие). Например, см. подписку на событие `ВариантыОтчетовПередУдалениемИдентификатораОбъектаМетаданных` в демонстрационной базе.

Для измерений регистров сведений с признаком `Ведущее` специальная обработка не требуется, поскольку при удалении ссылки все записи по этому измерению удаляются автоматически.

Отключение контроля использования удаляемых объектов при выполнении удаления

Для объектов, в которых допускается наличие битых ссылок и критично время записи при выполнении операции удаления, проверка использования удаляемых объектов отключается добавлением в коллекцию `ДополнительныеСвойства` свойства с именем `НеВыполнятьКонтрольУдаляемых`:

```
ДокументОбъект = Документ.ПолучитьОбъект();
ДокументОбъект.ДополнительныеСвойства.Вставить("НеВыполнятьКонтрольУдаляемых", Истина);
```

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы следует использовать следующие роли:

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Безвозвратное удаление объектов информационной базы.

Использование при разработке конфигурации

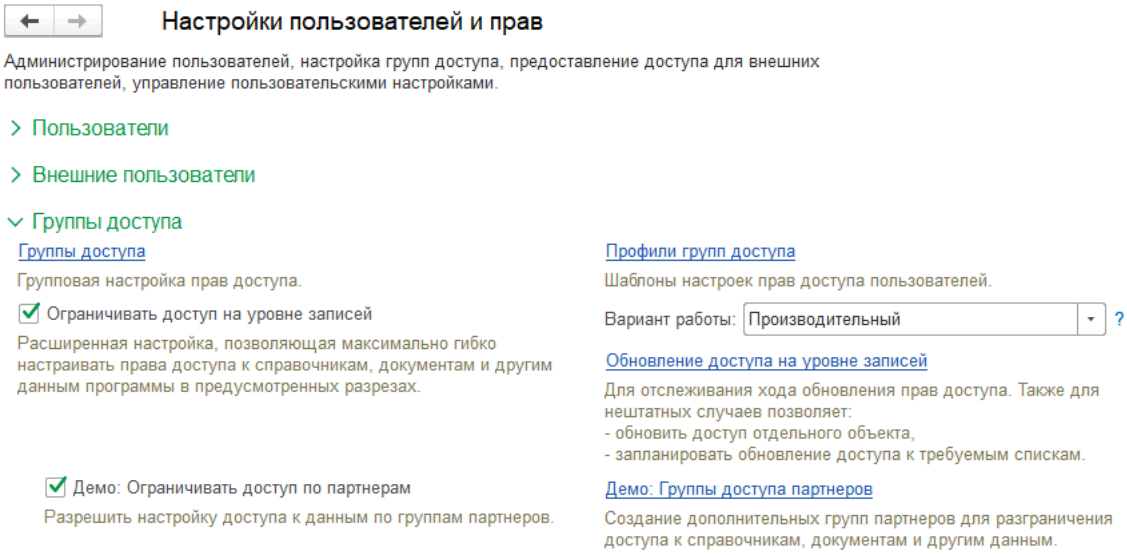
Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между различными приложениями. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Управление доступом

Подсистема **Управление доступом** позволяет настраивать права пользователей для произвольных элементов данных информационной базы (элементов справочников, документов, записей регистров, бизнес-процессов, задач и т. д.).

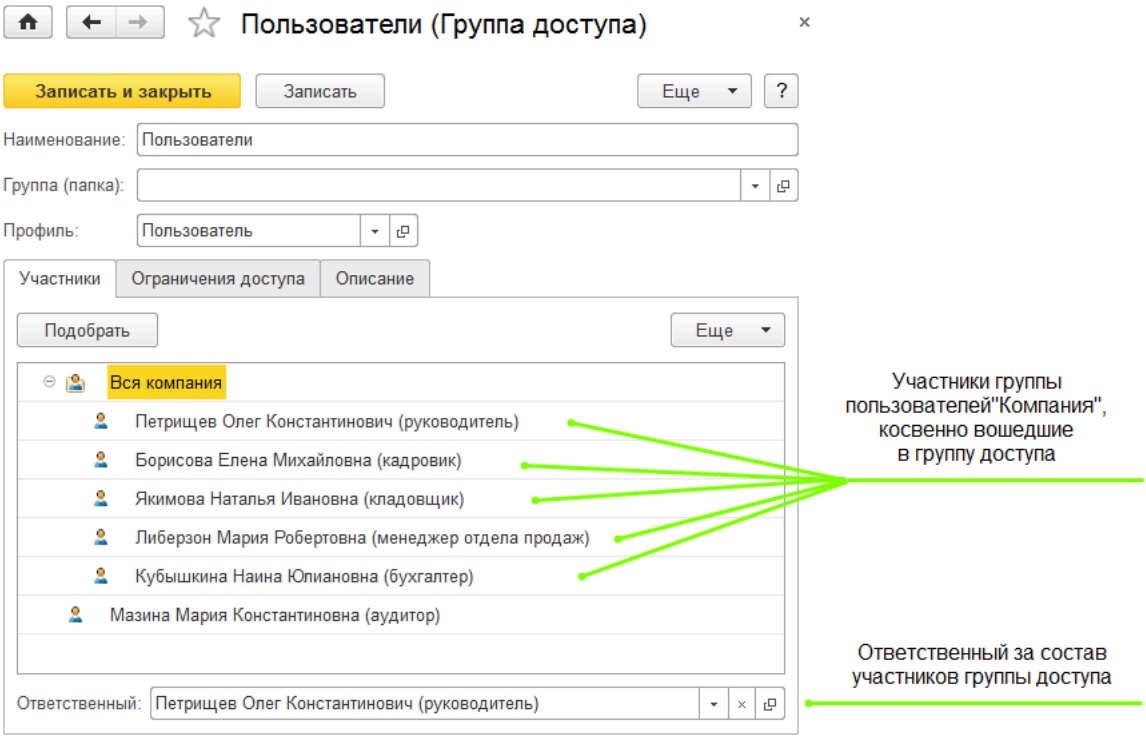


Возможно ограничение прав как для отдельных типов объектов метаданных, так и на уровне записей одного типа объекта. Кроме того, для отдельных объектов информационной базы возможна индивидуальная настройка прав доступа подобно папкам файлов операционной системы.

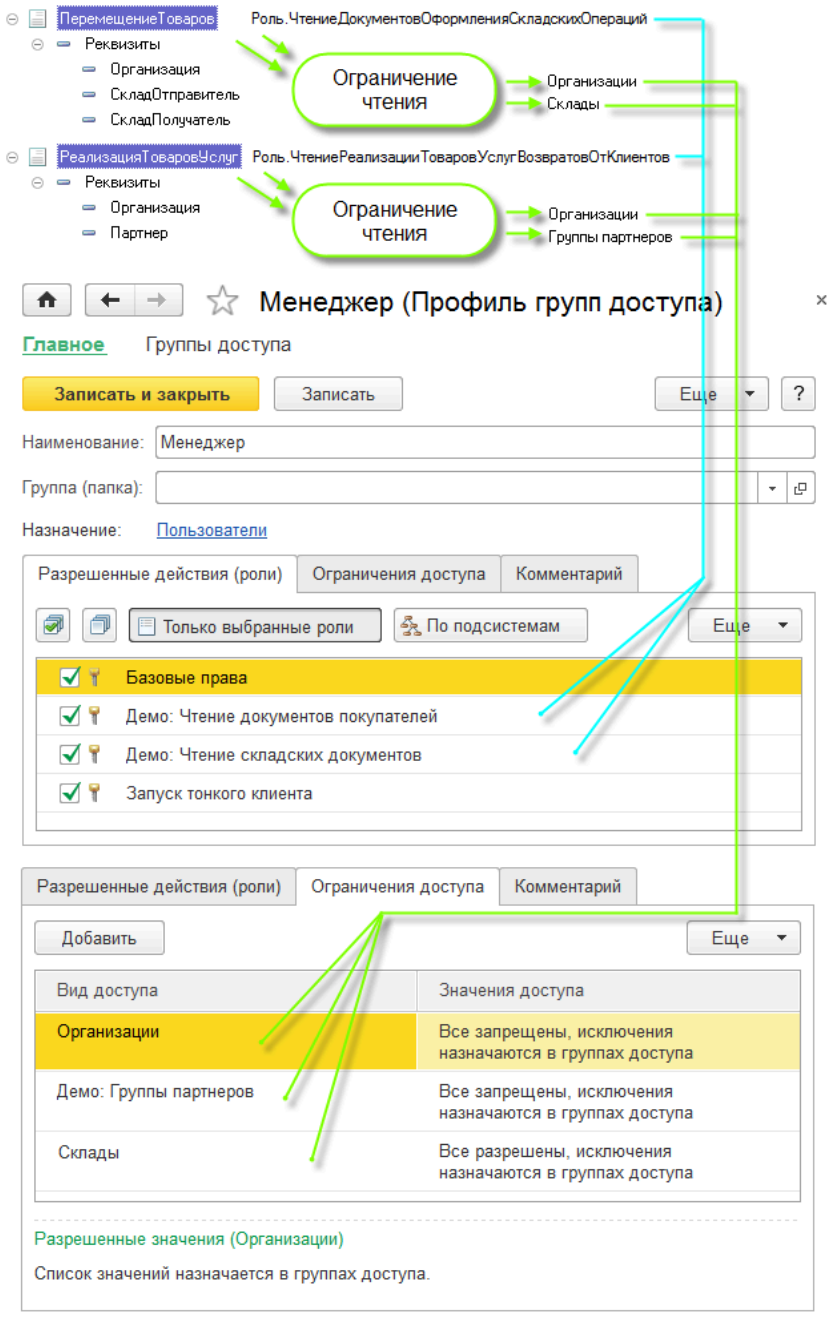
Для своей работы подсистема использует возможности подсистемы **Пользователи**.

Группы доступа и их профили

Для настройки прав доступа пользователей и групп пользователей предназначен справочник **Группы доступа**. Если для некоторых объектов информационной базы разрешен доступ с помощью внешних пользователей, то в группы доступа можно также помещать внешних пользователей и группы внешних пользователей.



Состав и логика ограничения прав доступа определяются профилем, заданным в группе доступа.



Профиль групп доступа представляет собой совокупность ролей (разрешенных действий) и видов доступа (в разрезе которых ограничивается доступ к данным информационной базы).

Вид доступа – это тип (или несколько типов) объектов информационной базы, в разрезе которых ограничиваются права доступа к данным информационной базы. Например, право чтения документов информационной базы может быть ограничено в разрезе определенных организаций, разрешенных для пользователя. В простейшем случае, если право чтения ограничено по виду доступа **Организации**, то будут прочитаны только те элементы данных (документы), у которых в поле **Организация** указана организация из списка разрешенных (или незапрещенных) организаций. Если же в группе доступа ограничение по виду доступа **Организации** не задано, то могут быть прочитаны все документы (как будто задан пустой список запрещенных организаций).

В самой группе доступа настраиваются списки разрешенных или запрещенных значений по видам доступа (например, списки организаций, групп партнеров, складов, групп номенклатуры и т. д.), которые используются в логике ограничения прав для объектов информационной базы.

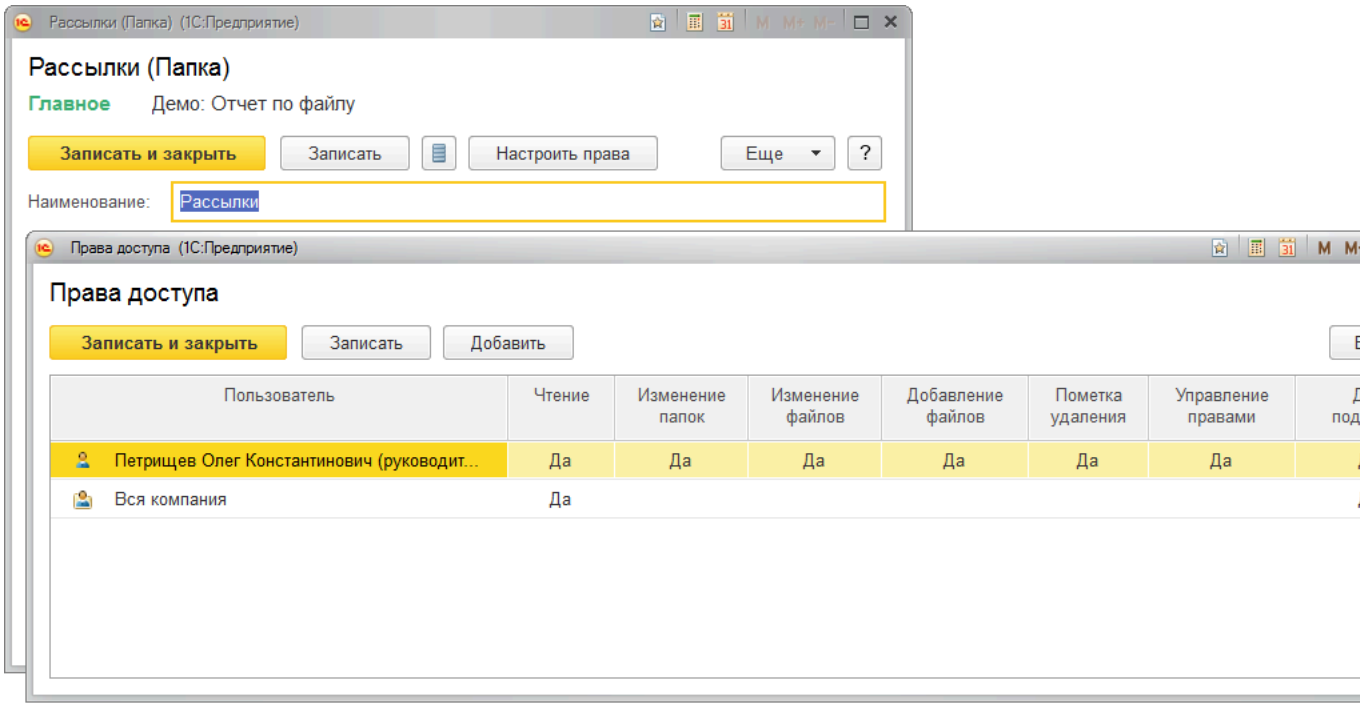
Таким образом, можно сказать, что у отдельного пользователя есть право доступа к объекту информационной базы, если пользователю разрешены связанные с этим объектом значения видов доступа.

См. также разделы Разработка ролей конфигурации и Разработка профилей групп доступа.

Настройка прав доступа к отдельным объектам информационной базы

Для отдельных объектов информационной базы возможна индивидуальная настройка прав доступа подобно папкам файлов операционной системы.

Например, для объектов справочника **Папки файлов** можно настроить доступ пользователей к отдельным папкам, а также к содержащимся в них дочерним папкам и элементам справочника **Файлы**. При этом индивидуальная настройка прав доступа к папкам возможна только для тех пользователей, которые входят в группу доступа пользователей со специальной ролью **Работа с папками файлов**.



Пример настройки групп доступа пользователей:

1. Профиль **Работа с папками файлов (дополнительно)** содержит роль **Работа с папками файлов**, которая предоставляет доступ к справочникам **Папки файлов** и **Файлы**.
2. Группа доступа **Работа с папками файлов (дополнительно)** содержит профиль **Работа с папками файлов (дополнительно)**.

Рекомендуется использовать одну группу доступа для всех пользователей. Тогда сочетание с другими группами доступа будет просто представить, т. к. в группе будут все права на папки, файлы и другие списки, доступ к которым зависит от настроек прав на папки файлов.

Настройка

1. Выбор производительного или стандартного варианта работы.
2. Выбор режима обычного или упрощенного интерфейса.
3. Описание логики ограничения прав и требуемых видов доступа.
4. Разработка видов доступа.
5. Разработка прав для настройки прав отдельных объектов.
6. Разработка ролей конфигурации.
7. Разработка профилей групп доступа.
8. Разработка ограничений прав доступа.
9. Конвертация ограничений прав доступа в стандартный вариант.
10. Настройка свойств объектов метаданных.
11. Создание описаний поставляемых профилей групп доступа для начального заполнения и обновления информационной базы.
12. Размещение в командном интерфейсе.
13. Настройка прав доступа пользователей.

Выбор производительного или стандартного варианта работы

Разработка ограничений доступа на уровне записей (RLS) возможна в двух вариантах: производительном или стандартном.

Производительный вариант основан на предварительном расчете прав доступа. Он позволяет достичь высокой производительности запросов с RLS, так как добавляет простой и статический фрагмент к текстам запросов в ролях. За счет этого он также обеспечивает одинаково хорошую скорость работы при различной прикладной логике ограничений доступа. В то же время предварительный расчет прав доступа занимает некоторое время, поэтому изменения в правах вступают в силу с некоторой задержкой (незначительной в большинстве случаев).

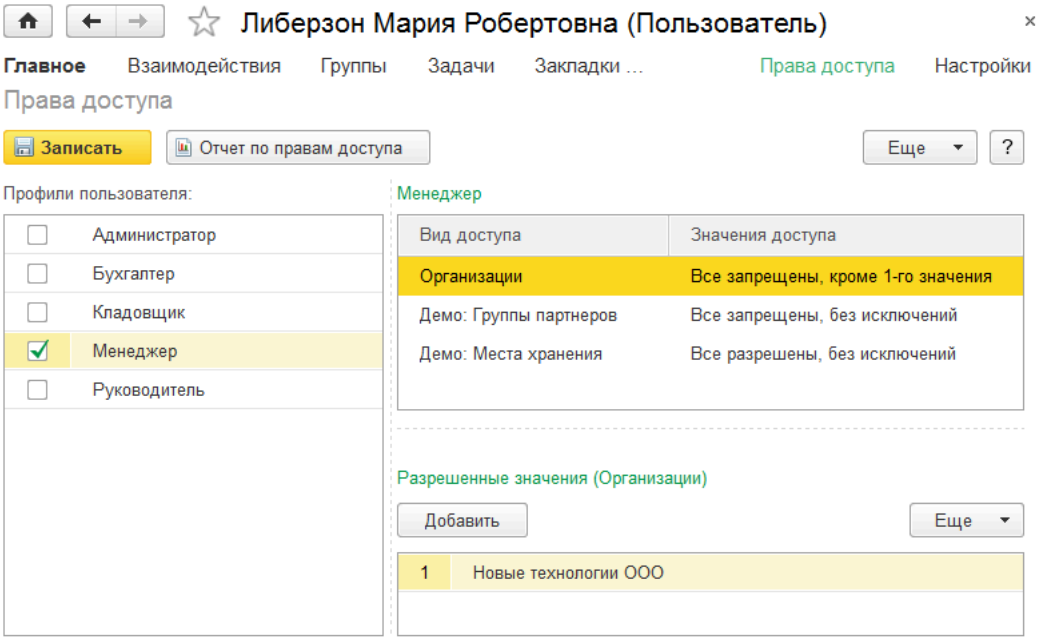
В отличие от этого, при использовании стандартного варианта RLS, расчет прав выполняется «на лету» непосредственно при доступе к данным. Соответственно, чем сложнее прикладная логика ограничения доступа, тем расчет прав выполняется медленнее, оказывая непосредственное влияние на скорость основных сценариев работы пользователей: просмотр списков, формирование отчетов, открытие документов.

Для вновь разрабатываемых прикладных решений рекомендуется производительный вариант работы. Для плавного перевода пользователей предыдущих версий прикладных решений со стандартного на производительный вариант рекомендуется некоторое время поддерживать сразу два варианта работы. Схема синхронной разработки подразумевает разработку ограничений доступа для производительного варианта и конвертацию их в стандартный вариант. Обеспечить синхронную разработку поможет отчет `ПроверкаВнедренияБСП.erf`, который проверяет корректность внедрения подсистемы для двух вариантов одновременно.

Выбор режима обычного или упрощенного интерфейса

Для конфигураций, предназначенных для небольшого количества пользователей (ориентировочно до 15), рекомендуется использовать упрощенный интерфейс настройки прав доступа. В таких организациях у сотрудников, как правило, уникальный набор прав доступа, который

настраивается индивидуально для каждого пользователя. В этом режиме пользователю ставится в соответствие один или несколько профилей, по каждому из которых можно настроить предусмотренные ограничения прав доступа.



Для организаций с большим количеством пользователей, напротив, лучше подходит обычный интерфейс групповой настройки прав доступа с помощью групп пользователей и групп доступа.

Для включения упрощенного интерфейса:

- изменить возвращаемое значение в переопределяемом модуле:

```
Процедура ПриОпределенииИнтерфейсаНастройкиДоступа(УпрощенныйИнтерфейс) Экспорт

    УпрощенныйИнтерфейс = Истина;

КонецПроцедуры
```

- в общей команде `ПраваДоступа` изменить свойство `Группа` с **Панель навигации формы.Перейти** на **Командная панель формы.Важное**.

В упрощенном интерфейсе также не следует размещать списки профилей и групп доступа в рабочем месте администратора.

Описание логики ограничения прав и требуемых видов доступа

При проектировании логики ограничений доступа рекомендуется придерживаться следующего подхода. Для каждого объекта метаданных предлагается сформулировать правила ограничения доступа:

- описать правила ограничения чтения;
- описать правила ограничения изменения и добавления.

При описании правил следует использовать определенные реквизиты объекта, содержащие ссылки на объекты (организации, контрагенты, далее – значения доступа), которые будут использоваться для отбора разрешенных объектов (например, документов). Следует уточнять, как именно должны работать отборы (далее – логика ограничения): требуется ли проверять, что, например, организация и контрагент разрешены для пользователя одновременно, или же достаточно того, что пользователю разрешена либо организация, либо контрагент.

Например, опишем логику ограничения доступа к документу **Перемещение товаров**, ответив на вопрос «Как нужно ограничить доступ к документу?»:

- Разрешить чтение документа пользователю, если ему разрешены указанные в документе организация и склад-отправитель или склад-получатель.
- Разрешить изменение и добавление документа пользователю, если ему разрешены организация и склад-отправитель и склад-получатель.
- Удаление всегда запрещено, кроме пользователя с ролью `ПолныеПрава`.

Отсюда следует, что доступ к документу должен ограничиваться в разрезе элементов справочников **Организации** и **Склады**. И, следовательно, в список видов доступа надо включить виды доступа **Организации** и **Склады**.

Виды доступа также могут быть групповыми. Это полезно, когда элементов в справочнике много, и поэтому доступ к данным должен ограничиваться не по отдельным элементам этого справочника, а по целым группам, сегментам или категориям. Например, доступ к документам должен настраиваться не в разрезе номенклатуры (это неудобно, т.к. их очень много в справочнике **Номенклатура**), а в разрезе групп номенклатуры (справочник `ГруппаДоступаНоменклатуры`). При чем, при необходимости, одна номенклатура также может входить в несколько таких групп, например, в бытовую химию и в продукты питания.

В результате мы составим полный список видов доступа и их свойств для всех объектов метаданных конфигурации. Кроме того, в состав подсистемы уже входят предопределенные виды доступа **Пользователи** и **Внешние пользователи**, подробнее см. в разделе

Предопределенные виды доступа.

Затем по сформулированным выше правилам следует задать в конфигурации эти виды доступа и разработать тексты ограничения доступа к объектам в модулях менеджеров этих объектов, о чем рассказывается далее в следующих разделах.

Разработка видов доступа

Для добавления в конфигурацию видов доступа необходимо задать их описание в процедуре `ПриЗаполненииВидовДоступа` общего модуля `УправлениеДоступомПереопределяемый`, а также расширить состав определяемых типов.

Например, код процедуры может выглядеть следующим образом:

```
Процедура ПриЗаполненииВидовДоступа(ВидыДоступа) Экспорт

    ВидДоступа = ВидыДоступа.Добавить();
    ВидДоступа.Имя = "МестаХранения";
    ВидДоступа.Представление = НСтр("ru = 'Места хранения'");
    ВидДоступа.ТипЗначений   = Тип("СправочникСсылка._ДемоМестаХранения");

    ВидДоступа = ВидыДоступа.Добавить();
    ВидДоступа.Имя = "ГруппыНоменклатуры";
    ВидДоступа.Представление    = НСтр("ru = 'Группы номенклатуры'");
    ВидДоступа.ТипЗначений      = Тип("СправочникСсылка._ДемоНоменклатура");
    ВидДоступа.ТипГруппЗначений = Тип("СправочникСсылка._ДемоГруппыДоступаНоменклатуры");
    УправлениеДоступом.ДобавитьДополнительныеТипыВидаДоступа(ВидДоступа,
        Тип("СправочникСсылка._ДемоВидыНоменклатуры"),
        Тип("СправочникСсылка._ДемоГруппыДоступаНоменклатуры"));

    ...
КонецПроцедуры
```

Вид доступа может быть связан с элементами справочника или перечисления. В этом примере вид доступа `МестаХранения` определяется элементами справочника `МестаХранения`, а для вида доступа `ГруппыНоменклатуры` указаны справочники `Номенклатура` и `ГруппаДоступаНоменклатуры`. Второй вариант подходит, когда доступ должен ограничиваться не по отдельным элементам справочника, а по их группам (сегментам, категориям).

Когда необходимо, чтобы у одного вида доступа было несколько типов значений, например, номенклатура и виды номенклатуры ограничиваются по одним и тем же группам доступа номенклатуры, они задаются с помощью процедуры `ДобавитьДополнительныеТипыВидаДоступа` общего модуля `УправлениеДоступом`.

Для групповых видов доступа также предусмотрено свойство `НесколькоГруппЗначений`. Если одно значение (например, номенклатура) может входить сразу в несколько групп, то следует установить это свойство в `Истина`. При этом следует учитывать особенность: если запрещена хотя бы одна из групп доступа номенклатуры, то при вычислении условия `ДляВсехСтрок(ЗначениеРазрешено(Товары.Номенклатура))` результат будет `Ложь` (доступ запрещен).

Для групповых видов доступа следует:

- создать дополнительный справочник для групп значений доступа (например, `ГруппыДоступаНоменклатуры`);
- если в свойстве `НесколькоГруппЗначений` указано `Ложь`, то создать реквизит `ГруппаДоступа` с типом групп значений (например, для номенклатуры с типом `ГруппыДоступаНоменклатуры`);
- в противном случае создать табличную часть `ГруппыДоступа` с реквизитом `ГруппаДоступа` (например, для номенклатуры с типом `ГруппыДоступаНоменклатуры`);
- добавленный реквизит `ГруппаДоступа` (или табличную часть `ГруппыДоступа` с реквизитом `ГруппаДоступа`) разместить в форме элемента значения доступа (например, справочника **Номенклатура**);
- создать функциональную опцию `ОграничиватьДоступНаУровнеЗаписей<ИмяПрикладногоРешения>`, например `ОграничиватьДоступНаУровнеЗаписейУправлениеТорговлей`, в ее свойстве `Хранение` задать константу `ОграничиватьДоступНаУровнеЗаписей` и включить в нее добавленные реквизиты (и табличные части).

В определяемые типы `ЗначениеДоступа` и `ЗначениеДоступаОбъект` добавить типы значений доступа и типы групп значений доступа.

> **Внимание!**
>
> Для того чтобы изменения вступили в силу сразу при разработке и отладке конфигурации, требуется обновить вспомогательные данные. См. раздел Обновление вспомогательных данных во время разработки.

Пример реализации процедуры `ПриЗаполненииВидовДоступа` см. в демонстрационной базе.

Использование видов доступа в зависимости от настроек приложения

Если в зависимости от функциональных опций или других настроек приложения должны меняться ограничения доступа к данным, то необходимо определить зависимости для видов доступа. Например, доступ к данным должен ограничиваться в разрезе мест хранения, только если в приложении включен функциональный блок учета товара на складах.

Для этого в процедуре `ПриЗаполненииИспользованияВидаДоступа` общего модуля `УправлениеДоступомПереопределяемый` указать зависимости, например, следующим образом:

```
// Заполняет использование видов доступа в зависимости от функциональных опций конфигурации,
// например "ИспользоватьГруппыДоступаНоменклатуры".
//
// Параметры:
//  ВидДоступа    - Строка - имя вида доступа заданное в процедуре ПриЗаполненииВидовДоступа.
//  Использование - Булево - начальное значение Истина.
//
Процедура ПриЗаполненииИспользованияВидаДоступа(ИмяВидаДоступа, Использование) Экспорт
    Если ИмяВидаДоступа = "_ДемоГруппыНоменклатуры" Тогда
        Использование = Константы._ДемоОграничиватьДоступПоНоменклатуре.Получить();

    ИначеЕсли ИмяВидаДоступа = "_ДемоГруппыПартнеров" Тогда
        Использование = Константы._ДемоОграничиватьДоступПоПартнерам.Получить();

    ИначеЕсли ИмяВидаДоступа = "_ДемоФизическиеЛица" Тогда
        Использование = Константы._ДемоОграничиватьДоступПоФизическимЛицам.Получить();
    КонецЕсли;
КонецПроцедуры
```

При изменении использования вида доступа требуется вызвать обновление разрешенных значений – например, из модуля менеджера значения константы:

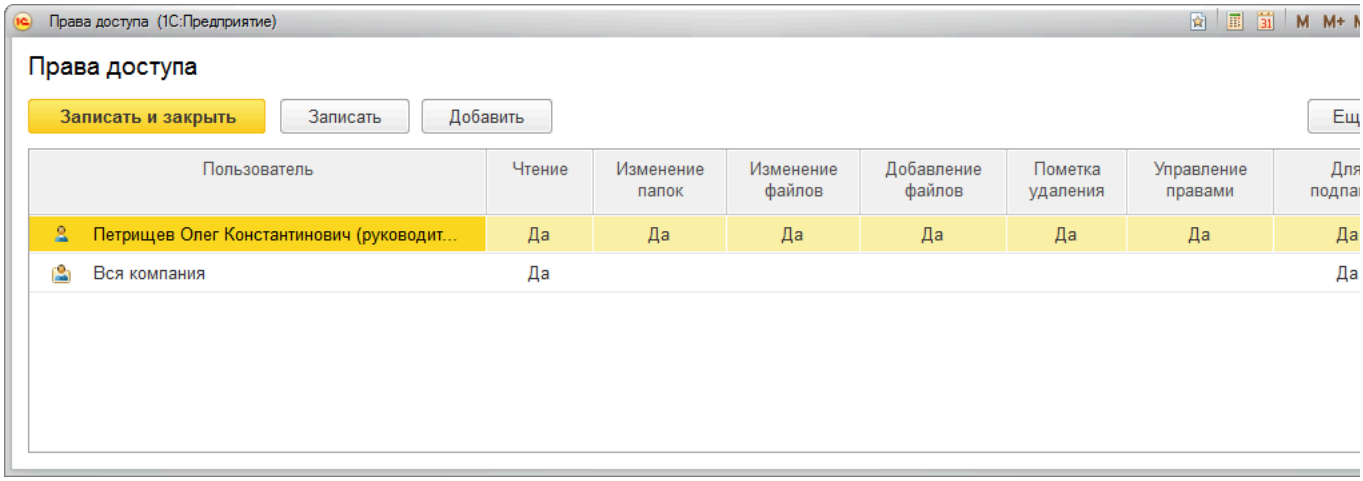
```
Процедура ПриЗаписи(Отказ)

    Если ОбменДанными.Загрузка Тогда
        Возврат;
    КонецЕсли;

    УправлениеДоступом.ОбновитьРазрешенныеЗначенияПриИзмененииИспользованияВидовДоступа();

КонецПроцедуры
```

В стандартном варианте RLS для изменения действующего значения в ограничениях доступа к объектам требуется перезапуск сеанса. В производительном варианте RLS перезапуск сеанса требуется только при изменении состава ролей пользователя информационной базы (как и для платформы **1С:Предприятия**).

Пример реализации процедуры `ПриЗаполненииИспользованияВидаДоступа` см. в демонстрационной базе.

Разработка прав для настройки прав отдельных объектов

В случае необходимости настройки прав для отдельных объектов информационной базы следует определить список типов этих объектов и в модуле `УправлениеДоступомПереопределяемый` и для каждого типа вписать права в процедуру `ПриЗаполненииВозможныхПравДляНастройкиПравОбъектов`, которые используются для автоматического создания формы настройки этих прав.



Например, для получения состава прав (колонок), как на рисунке, требуется следующий код:

```
Процедура ПриЗаполненииВозможныхПравДляНастройкиПравОбъектов(ВозможныеПрава) Экспорт

    ////////////////////////////////////////////////////////////
    // Справочник.ПапкиФайлов

    // Право "Чтение папок и файлов".
    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "Чтение";
    Право.Заголовок     = НСтр("ru = 'Чтение'");
    Право.Подсказка     = НСтр("ru = 'Чтение папок и файлов'");
    Право.НачальноеЗначение = Истина;
    // Права для стандартных шаблонов ограничений доступа.
    Право.ЧтениеВТаблицах.Добавить("*");

    // Право "Изменение папок"
    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "ИзменениеПапок";
    Право.Заголовок     = НСтр("ru = 'Изменение
                               |папок'");
    Право.Подсказка     = НСтр("ru = 'Добавление, изменение и
                               |пометка удаления папок файлов'");
    // Права, требуемые для этого права.
    Право.ТребуемыеПрава.Добавить("Чтение");
    // Права для стандартных шаблонов ограничений доступа.
    Право.ИзменениеВТаблицах.Добавить(Метаданные.Справочники.ПапкиФайлов.ПолноеИмя());

    // Право "Изменение файлов"
    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "ИзменениеФайлов";
    Право.Заголовок     = НСтр("ru = 'Изменение
                               |файлов'");
    Право.Подсказка     = НСтр("ru = 'Изменение файлов в папке'");
    // Права, требуемые для этого права.
    Право.ТребуемыеПрава.Добавить("Чтение");
    // Права для стандартных шаблонов ограничений доступа.
    Право.ИзменениеВТаблицах.Добавить("*");

    // Право "Добавление файлов"
    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "ДобавлениеФайлов";
    Право.Заголовок     = НСтр("ru = 'Добавление
                               |файлов'");
    Право.Подсказка     = НСтр("ru = 'Добавление файлов в папку'");
    // Права, требуемые для этого права.
    Право.ТребуемыеПрава.Добавить("ИзменениеФайлов");

    // Право "Пометка удаления файлов".
    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "ПометкаУдаленияФайлов";
    Право.Заголовок     = НСтр("ru = 'Пометка
                               |удаления'");
    Право.Подсказка     = НСтр("ru = 'Пометка удаления файлов в папке'");
    // Права, требуемые для этого права.
    Право.ТребуемыеПрава.Добавить("ИзменениеФайлов");

    Право = ВозможныеПрава.Добавить();
    Право.ВладелецПрав  = Метаданные.Справочники.ПапкиФайлов.ПолноеИмя();
    Право.Имя           = "УправлениеПравами";
    Право.Заголовок     = НСтр("ru = 'Управление
                               |правами'");
    Право.Подсказка     = НСтр("ru = 'Управление правами папки'");
    // Права, требуемые для этого права.
    Право.ТребуемыеПрава.Добавить("Чтение");
КонецПроцедуры
```

- Указать состав типов как состав владельцев прав в процедуре `ПриЗаполненииВозможныхПравДляНастройкиПравОбъектов`:
  - в определяемом типе `ВладелецНастроекПрав`,
  - в определяемом типе `ВладелецНастроекПравОбъект`,
  - в общей команде `НастроитьПрава`.
- Использовать в модуле `УправлениеДоступом` функцию `ЕстьПраво` для получения пользователем разрешения на права, описанные в переопределяемой процедуре (выше).
- Проверка прав Чтение, Изменение возможна в текстах ограничения с помощью функций `ЧтениеОбъектаРазрешено`, `ИзменениеОбъектаРазрешено` (в стандартном варианте с помощью специального вида доступа `НастройкиПрав`).

> **Внимание!**
>
> Для того чтобы изменения вступили в силу сразу при разработке и отладке конфигурации, требуется обновить вспомогательные данные. См. раздел Обновление вспомогательных данных во время разработки.

Пример реализации процедуры `ПриЗаполненииВозможныхПравДляНастройкиПравОбъектов` см. в демонстрационной базе.

Разработка ролей конфигурации

Методические рекомендации по проектированию состава ролей конфигурации см. в разделе Базовая функциональность.

Для каждой спроектированной роли нужно:

- добавить в конфигурацию объект метаданных «роль», установить имя и синоним;
- снять флажок **Устанавливать права для новых объектов**;
- снять флажок **Независимые права подчиненных объектов**;
- установить флажки напротив требуемых прав по каждому объекту метаданных конфигурации;
- для каждого права каждого объекта метаданных указать стандартные шаблоны ограничения доступа. Применяемые в роли стандартные шаблоны следует предварительно скопировать из поставляемой роли `ИзменениеУчастниковГруппДоступа`.
  - Логика ограничения реализуется в процедурах `ПриЗаполненииОграниченияДоступа` модулей менеджеров объектов метаданных, а в правах ролей указываются шаблоны `ДляОбъекта`, `ДляРегистра` и их параметры, которые вычисляются строго на основе указанной логики ограничения (подробнее ниже).
- В стандартном варианте логика ограничения реализуется через параметры стандартных шаблонов `ПоЗначениям`, `ПоЗначениямРасширенный`, `ПоЗначениямИНаборамРасширенный`, `ПоЗначениямИНаборамРасширенный`.
  - При установке ограничений прав следует указывать их для всех полей (в колонке **Поля** выбирать вариант **Прочие поля** без указания ограничений для конкретных полей). Это обусловлено тем, что для пользователя состав строк в списках и отчетах не должен зависеть от состава колонок, которые он настроил для отображения тем или иным способом.
    - В частности, безусловно запрещенные поля (например, для всех, кроме администратора), следует вынести в отдельную таблицу, на которую назначить нужные права.
    - Безусловно разрешенные поля делать не требуется (ранее для работы механизмов **1С:Предприятия** требовалось предоставить доступ к некоторым стандартным реквизитам, например Код, но более этого не требуется, так как теперь **1С:Предприятие** использует для технологического доступа привилегированный режим).

Для того чтобы исключить снижение производительности, при наличии в конфигурации нескольких ролей, предоставляющих доступ к одному объекту метаданных, следует указывать в таких ролях одинаковые тексты ограничений у одинаковых прав доступа (см. стандарт Эффективные условия запросов, пункт 7).

При необходимости переименовать роль следует учитывать методику раздела Хранение ссылок на объекты метаданных.

Разработка профилей групп доступа

Состав профилей групп доступа рекомендуется определять, исходя из должностных обязанностей сотрудников, работающих в приложении. Например, для бухгалтерских приложений это могут быть профили **Бухгалтер** и **Главный бухгалтер**, отличающиеся уровнем полномочий, для решений в области торговли - **Менеджер по продажам**, **Руководитель отдела продаж** и т.п.

В каждом прикладном решении также предусмотрен поставляемый профиль **Администратор**, который дает неограниченный доступ к любым данным и операциям. При работе в модели сервиса он позволяет администрировать только свою область данных, но не всю информационную базу. В однопользовательских решениях он соответствует обычным полномочиям пользователя.

Затем в режиме **1С:Предприятие** необходимо создать требуемое количество профилей групп доступа пользователей, заполнив их ролями и видами доступа, созданными на предыдущем шаге.

> **Важно!**
>
> В профиль следует включать только те виды доступа, которые задействованы в ограничениях прав на таблицы, настроенные в ролях, выбранных для профиля. Настройка видов доступа в профиле и группе доступа не влияет на ограничение прав на таблицы, в ограничении которых виды доступа не задействованы.

В профиле любой вид доступа можно сделать предустановленным, если это необходимо для единообразной настройки всех групп доступа этого профиля. Например, вид доступа, используемый для «разрезания» таблиц по признаку `ХозяйственнаяОперация`, удобнее сделать предустановленным. Для кассира – свой список хозяйственных операций, для подотчетного лица – свой, а для бухгалтера – список всех хозяйственных операций по отношению, например, к документам **Расходный кассовый ордер**.

Затем вписать в процедуру `ПриЗаполненииПоставляемыхПрофилейГруппДоступа` общего модуля `УправлениеДоступомПереопределяемый` код по созданию и обновлению поставляемых профилей групп доступа при начальном заполнении и обновлении информационной базы. Для автоматического формирования текста этой процедуры следует применять обработку `УправлениеДоступом.epf`, которая формирует код на основании профилей групп доступа, существующих в информационной базе (см. также раздел Создание описаний поставляемых профилей групп доступа для начального заполнения и обновления информационной базы. Пример реализации процедуры `ПриЗаполненииПоставляемыхПрофилейГруппДоступа` см. в демонстрационной базе.

Разработка ограничений прав доступа

Разработка ограничений состоит из разработки текста логики ограничения в процедурах `ПриЗаполненииОграниченияДоступа` модулей менеджеров объектов и автоматического вычисления (на основе указанной логики) варианта шаблона для вставки в роли и состава типов определяемых типов - см. ниже Типовые случаи логики ограничения и Вставка ограничений в права ролей.

- Чтобы ограничение работало, объект должен быть подключен к подсистеме (см. ниже Подключение объектов к подсистеме).
- Требуемый вариант шаблона с параметрами, а также состав типов определяемых типов показывает инструмент разработчика `УправлениеДоступом.epf`.
- Если указан неправильный вариант шаблона или его параметры, тогда в режиме **1С:Предприятия** возникнет ошибка выполнения запроса с RLS. Если не хватает типов в определяемых типах, тогда ошибка возникнет при обновлении доступа. Отчет `ПроверкаВнедренияБСП.erf` умеет находить такие ошибки, а также автоматически исправлять их.
- При изменении логики ограничения рекомендуется запускать отчет `ПроверкаВнедренияБСП.erf` (с флажком **Исправлять ошибки** и с отбором по подсистеме **Управление доступом**) для обновления ограничений в ролях, а также определяемых типов и предопределенных элементов в справочнике `ИдентификаторыОбъектовМетаданных`.

> **Примечание**
>
> Рекомендуется добавлять в справочник `ИдентификаторыОбъектовМетаданных` предопределенные элементы всех регистров, чтобы при изменении ограничений доступа в модулях менеджеров не требовалось добавления предопределенных элементов при доработке конфигурации на внедрении, а также при использовании расширений конфигурации.

Подключение объектов к подсистеме

Все объекты метаданных, для которых описана логика ограничения, необходимо указать в процедуре `ПриЗаполненииСписковСОграничениемДоступа` общего модуля `УправлениеДоступомПереопределяемый`.

Для выбранных объектов метаданных необходимо вставить в модуль формы объекта в процедуру обработчика события `ПриЧтенииНаСервере` вызов:

```
УправлениеДоступом.ПриЧтенииНаСервере(ЭтотОбъект, ТекущийОбъект);
```

При выполнении вызова будет проверено право изменения объекта, и если права изменения нет, то свойство формы `ТолькоПросмотр` будет установлено `Истина`.

Кроме того, в модуль формы объекта в процедуру обработчика события `ПослеЗаписиНаСервере` необходимо вставить вызов:

```
УправлениеДоступом.ПослеЗаписиНаСервере(ЭтотОбъект, ТекущийОбъект, ПараметрыЗаписи);
```

При выполнении вызова будет выполнен запуск обновления доступа, если обновление доступа запланировано, но еще не выполняется, что ускоряет расчет прав для зависимых объектов (например, в ограничении которых есть обращение «через точку» или соединения или одна из функций `ЧтениеОбъектаРазрешено`, `ИзменениеОбъектаРазрешено`).

Вставка ограничений в права ролей

После подключения объекта к подсистеме нужно вручную разово вставить ограничение в права ролей. Если этого не сделать (оставить текст ограничения права в роли пустым), тогда считается, что эта роль не ограничивает право (вариант роли `БезОграничения`) и отчет `ПроверкаВнедренияБСП.erf` не покажет, что в роли в ограничении права есть ошибка, и соответственно не обновит его, когда это требуется после внесения изменений в логику ограничения. Ограничения права имеет обязательный формат:

```
#Если &ОграничениеДоступаНаУровнеЗаписейУниверсально #Тогда
<шаблон ограничения с параметрами для производительного варианта>
#Иначе
<шаблон ограничения с параметрами для стандартного варианта>
#КонецЕсли
```

Если стандартный вариант работы не поддерживается, тогда вместо шаблона ограничения для поддержки стандартного варианта следует написать `ГДЕ ЛОЖЬ`, например:

```
#Если &ОграничениеДоступаНаУровнеЗаписейУниверсально #Тогда
#ДляОбъекта("")
#Иначе
ГДЕ ЛОЖЬ
#КонецЕсли
```

Ограничения в ролях реализуются с помощью одного из двух шаблонов:

```
#ДляОбъекта("<Необязательное имя поля объекта-владельца>")
#ДляРегистра("<Описание регистра>", "<Имя поля 1>", "<Необязательное имя поля 2>", "<Необязательное имя поля 3>", "<Необязательное имя поля 4>"
```

Вариант шаблона и его параметры вычисляются автоматически на основе описания ограничения в модулях менеджеров.

При компиляции инструкций препроцессора шаблона происходит проверка правильности используемого шаблона и его параметров (через параметры сеанса). Если вариант шаблона или его параметры указаны неправильно, тогда возникнет ошибка. Например, если указать ограничение

```
#ДляОбъекта("Организация")
```

для права Чтение в роли для списка `Документ._ДемоЗаказПокупателя`, тогда при попытке открыть список под пользователем с ограниченными правами возникнет ошибка:

```
Ошибка<<?>>: Требуется актуализировать ограничение доступа по причине: Не удалось определить вариант ограничения доступа в параметрах сеанса для
```

Результат вычисления можно получить с помощью инструмента разработчика `УправлениеДоступом.epf`, который удобно использовать для разработки ограничений. Инструмент формирует пошаговую инструкцию с фрагментами текста и составом определяемых типов для вставки в требуемые места вручную.

Например, часть инструкции:

```
    - в роли с назначением для пользователей на права Чтение, Добавление, Изменение
      объекта метаданных РегистрыСведений._ДемоРаботникиОрганизаций установить ограничение
      (и добавить соответствующий шаблон ограничения, если его еще нет в роли):

#Если &ОграничениеДоступаНаУровнеЗаписейУниверсально #Тогда
#ДляРегистра("ИдентификаторыОбъектовМетаданных.РегистрСведений_ДемоРаботникиОрганизаций", "Организация", "ФизическоеЛицо", "", "", "")
#Иначе
#ПоЗначениям( "РегистрСведений._ДемоРаботникиОрганизаций", "", "",
"_ДемоОрганизации", "Организация",
"_ДемоФизическиеЛица","ФизическоеЛицо", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
#КонецЕсли
```

Типовые случаи логики ограничения

Ниже даны описания логики ограничения в процедуре `ПриЗаполненииОграниченияДоступа` на простых примерах.

- Организация и контрагент в шапке документа:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Контрагент)";
```

- Организация в шапке документа, контрагент в табличной части, достаточно одного разрешенного контрагента:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент)";
```

- Организация в шапке документа, контрагент в табличной части, достаточно одного разрешенного контрагента (если табличная часть пустая, тогда доступ по контрагенту разрешен):

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент, Null КАК Истина)";
```

- Организация в шапке документа, контрагент в табличной части, и требуется, чтобы все контрагенты были разрешены (если табличная часть пустая, тогда доступ по контрагенту запрещен):

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ДляВсехСтрок(ЗначениеРазрешено(Поставщики.Контрагент))";
```

- Организация и контрагент в табличной части, при этом достаточно, чтобы любая из пар организации с контрагентом была разрешена:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Поставщики.Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент)";
```

- Организация и контрагент в табличной части, при этом требуется, чтобы все пары организации с контрагентом были разрешены:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ДляВсехСтрок(
|      ЗначениеРазрешено(Поставщики.Организация)
|    И ЗначениеРазрешено(Поставщики.Контрагент))";
```

- Организация и контрагент в табличной части, при этом требуется, чтобы одна из организаций и один из контрагентов были разрешены:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ДляОднойИзСтрок(ЗначениеРазрешено(Поставщики.Организация))
|И ДляОднойИзСтрок(ЗначениеРазрешено(Поставщики.Контрагент))";
```

- Отправитель – измерение составного типа, при этом требуется проверять только ссылки `Справочник.Склады`:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Отправитель ТОЛЬКО Справочник.Склады)";
```

- Организация в шапке (Владелец), головная организация в справочнике организаций, при этом требуется, чтобы изменение было разрешено, когда разрешена организация в шапке, а чтение было разрешено:
  - когда разрешена организация в шапке (Владелец),
  - когда разрешена непустая головная организация организации в шапке,
  - когда разрешена организация, у которой организация в шапке (Владелец) является головной.

```
Ограничение.Текст =
"ПрисоединитьДополнительныеТаблицы
|ЭтотСписок КАК ЭтотСписок
|
|ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК Владельцы
|    ПО Владельцы.Ссылка = ЭтотСписок.Владелец
|
|ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК ОбособленныеПодразделения
|    ПО ОбособленныеПодразделения.ГоловнаяОрганизация = Владельцы.Ссылка
|;
|РазрешитьЧтение
|ГДЕ
|    ЗначениеРазрешено(Владелец)
| ИЛИ ЗначениеРазрешено(Владельцы.ГоловнаяОрганизация, ПустаяСсылка КАК Ложь)
| ИЛИ ЗначениеРазрешено(ОбособленныеПодразделения.Ссылка)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ЗначениеРазрешено(Владелец)";
```

- Перенос прав от объекта-владельца (например, на присоединенные файлы).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(ВладелецФайла)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(ВладелецФайла)";
```

- Ограничение по чтению владельца (например, журнал документов).

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|    ЧтениеОбъектаРазрешено(Ссылка)";
```

- Проверка настроек прав Чтение и Изменение на объекты (например, настройки прав на элементы справочника `ПапкиФайлов`).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(Ссылка)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(Ссылка)";
```

- Проверка настроек прав Чтение и Изменение на дополнительные объекты (например, настройки прав на справочник `Файлы`, сделанные по элементам справочника `ПапкиФайлов`).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(ВладелецФайла)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(ВладелецФайла)";
```

- Учетная запись и контрагент в шапке, при этом достаточно, чтобы было разрешение для одного из них. При этом если отключено ограничение по учетным записям, должно остаться ограничение по контрагентам, а если отключено ограничение по контрагентам, остается ограничение по учетным записям. Если оба ограничения отключены, то доступ должен быть разрешен.

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
| ЗначениеРазрешено(УчетнаяЗапись, Отключено КАК Ложь)
| ИЛИ ЗначениеРазрешено(Контрагент, Отключено КАК Ложь)";
```

- Проверка прав доступа на объекты метаданных, связанные прикладной логикой.

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
| ЗначениеРазрешено(Организация)
|   И (ПравоДоступа(Просмотр, Документ.СчетНаОплатуПокупателю)
|      ИЛИ ПравоДоступа(Просмотр, ПланСчетов.Хозрасчетный)
|      ИЛИ ПравоДоступа(Использование, HTTPСервис.ExternalAPI.ШаблонURL.ПоказателиМониторыРуководителя.Метод.Получить)
|        И РазделМониторыПубликуемый)";
```

- Проверка роли, используемой в качестве дополнительного права. Например, когда создана роль `ДобавлениеИзменениеДокументовПокупателейБезПроведения`, выполняется проверка, что у пользователя есть роль разрешающая проведение, если документ не проведен.

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
| ЗначениеРазрешено(Организация)
|   И ЗначениеРазрешено(Партнер)
|   И (НЕ Проведен ИЛИ РольДоступна(ДобавлениеИзменениеДокументовПокупателей))";
```

- Разрешить доступ, если в реквизите `Организация` содержится разрешенное значение или пустая ссылка:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|   ЗначениеРазрешено(Организация, ПустаяСсылка КАК Истина)";
```

- Разрешить доступ, если в реквизите `Контрагент` табличной части `Поставщики` содержится разрешенное значение или пустая ссылка, а сама табличная часть при этом непустая:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|   ЗначениеРазрешено(Поставщики.Контрагент, ПустаяСсылка КАК Истина)";
```

- Разрешить доступ, если реквизит `Контрагент` табличной части `Поставщики` содержит разрешенное значение, пустую или несуществующую ссылку, или же сама табличная часть пустая:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|   ЗначениеРазрешено(Поставщики.Контрагент, ПустаяСсылка КАК Истина, Null КАК Истина)";
```

- Разрешить доступ, если реквизит "через точку" (`Склад` или `Организация`) содержит разрешенное значение или реквизит "слева от точки" (`Владелец`, `Дополнение`, `КлючАналитики`) содержит пустую или несуществующую ссылку:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|     ЗначениеРазрешено(Владелец.Склад, Null КАК Истина)
| Или ЗначениеРазрешено(Дополнение.КлючАналитики.Организация, Null КАК Истина)";
```

- Разрешить доступ, если в реквизите `Заказ` содержится ссылка на документ с правом чтения, либо пустая или несуществующая ссылка:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|   ЧтениеОбъектаРазрешено(Заказ, Null КАК Истина)";
```

Синтаксис логики ограничения

Структура всех возможных вариантов описания логики ограничения в процедуре `ПриЗаполненииОграниченияДоступа`:

```
Ограничение.Текст =
"ПрисоединитьДополнительныеТаблицы
|ЭтотСписок КАК <Псевдоним>
|ЛЕВОЕ СОЕДИНЕНИЕ <Любая таблица> КАК <Псевдоним>
|ПО <Простое условие>
|… <Другие соединения>
|;
|РазрешитьЧтение
|ГДЕ
|        <Условие>
|  И/ИЛИ НЕ <Условие>
|  И/ИЛИ ВЫБОР
|          КОГДА <Условие>
|            ТОГДА <Условие>
|            ИНАЧЕ <Условие>
|         КОНЕЦ
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|   <Условие> …";

Ограничение.ТекстДляВнешнихПользователей =
"<аналогично тексту для пользователей>";
```

Условие может быть в виде специальных функций ограничения:

- `ЗначениеРазрешено` (<Реквизит> [<проверяемые типы>] [, <уточнение сравнения 1> [, <уточнение сравнения 2>] …]), где проверяемые типы могут быть (значения непроверяемых типов запрещены, если не уточнены отдельно):
  - `ТОЛЬКО` <Имя таблицы>;
  - `ТОЛЬКО` (<Имя таблицы 1>, <Имя таблицы 2>, …);
  - `КРОМЕ` <Имя таблицы>;
  - `КРОМЕ` (<Имя таблицы 1>, <Имя таблицы 2>, …);
- Уточнение сравнения может быть:
  - `ПустаяСсылка КАК Ложь` / `Истина`;
  - `Неопределено КАК Ложь` / `Истина`;
  - `Null КАК Ложь` / `Истина`;
  - `Отключено КАК Ложь` / `Истина` (только для функции `ЗначениеРазрешено`);
  - `<Таблица> КАК Ложь` / `Истина` (например, `Справочник.Проекты КАК Истина`).
- `ЧтениеОбъектаРазрешено` (так же, как для функции `ЗначениеРазрешено`).
- `ЧтениеСпискаРазрешено` (так же, как для функции `ЗначениеРазрешено`).
- `ИзменениеОбъектаРазрешено` (так же, как для функции `ЗначениеРазрешено`).
- `ИзменениеСпискаРазрешено` (так же, как для функции `ЗначениеРазрешено`).
- Последние 2 функции можно указать только в разделе `РазрешитьИзменениеЕслиРазрешеноЧтение`.
- `ЭтоАвторизованныйПользователь` (так же, как для функции `ЗначениеРазрешено`).
- `ПравоДоступа` (<Имя права доступа>, <Полное имя объекта метаданных>) – проверяет право в ролях профиля группы доступа, параметры аналогичны одноименному методу платформы.
- `РольДоступна` (<Имя роли>) - проверяет наличие роли в профиле группы доступа.

Пример того, как выглядит использование расширенных свойств сравнения в специальных функциях:

- `ЗначениеРазрешено(Владелец ТОЛЬКО Справочник.Организации)`.
- `ЗначениеРазрешено(Владелец КРОМЕ (Справочник.Проекты, Справочник.Номенклатура), ПустаяСсылка КАК Истина)`.
- `ЗначениеРазрешено(Владелец, Неопределено КАК Истина)`.
- `ПравоДоступа(Использование, Метаданные.HTTPСервисы.ExternalAPI.ШаблоныURL.ПоказателиМонитораРуководителя.Методы.Получить)`.
- `РольДоступна(ПроведениеДокументовПокупателей)`.

Условие предоставляет следующие стандартные виды сравнения:

- <Реквизит> `Равно` | `Не равно` <предопределенное значение>;

- <Реквизит> `В списке` | `Не в списке` <предопределенное значение 1>, …
- <Реквизит> `ЕСТЬ NULL`;
- `ТипЗначения` (<Реквизит>) = `Тип` (<Имя таблицы>).

Пример того, как выглядит использование стандартных видов сравнения:

- `ВидПоставщика = Значение(Перечисление.ВидыПоставщиков.Основной)`;
- `ВидПоставщика В (Значение(Перечисление.ВидыПоставщиков.Основной), Значение(Перечисление.ВидыПоставщиков.Дополнительный))`.

Условие также представлено двумя модификаторами, которые указывают способ сложения результатов проверок (столбец результатов), выполненных в условии, если использовался реквизит табличной части или дополнительной таблицы, который может содержать несколько значений (столбец значений):

- `ДляВсехСтрок` (<Условие>) – объединить результаты проверок в строках с помощью логического «И».
- `ДляОднойИзСтрок` (<Условие>) – объединить результаты проверок в строках с помощью логического «ИЛИ» (это поведение по умолчанию, но есть случаи, когда поведения по умолчанию недостаточно).

Синтаксис в зависимости от наличия ограничений по правам.

- Нет ограничения чтения и нет ограничения изменения:

```
Ограничение.Текст =
"";
```

- Одинаковые ограничения чтения и изменения:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ ... ";
```

- Разные ограничения чтения и изменения:

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ ...
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ ... ";
```

- Нет ограничения чтения, а есть только ограничение изменения:

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ ИСТИНА
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ ... ";
```

Синонимы ключевых слов, используемых в ограничении доступа:

Ключевое слово на русском языке	Ключевое слово на английском языке
• `ПрисоединитьДополнительныеТаблицы` • `ЭтотСписок` • `РазрешитьЧтениеИзменение` • `РазрешитьЧтение` • `РазрешитьИзменениеЕслиРазрешеноЧтение` • `ЗначениеРазрешено` • `ЧтениеОбъектаРазрешено` • `ИзменениеОбъектаРазрешено` • `ЧтениеСпискаРазрешено` • `ИзменениеСпискаРазрешено` • `ЭтоАвторизованныйПользователь` • `ДляВсехСтрок` • `ДляОднойИзСтрок` • `Отключено` • `Кроме` • `Только`	• `AttachAdditionalTables` • `ThisList` • `AllowReadUpdate` • `AllowRead` • `AllowUpdateIfReadingAllowed` • `ValueAllowed` • `ObjectReadingAllowed` • `ObjectUpdateAllowed` • `ListReadingAllowed` • `ListUpdateAllowed` • `IsAuthorizedUser` • `ForAllRows` • `ForAtLeastOneRow` • `Disabled` • `Except` • `Only`

Применение и работа функций ограничения доступа

Функцию `ЗначениеРазрешено` следует использовать, как основной способ построения ограничения доступа.

- Функция выполняет поиск проверяемого значения среди разрешенных значений в группах доступа, профили которых предоставляют соответствующие права на список. Поиск осуществляется только для значений доступа (типы значений доступа указаны в свойствах `ТипЗначений` видов доступа в процедуре `ПриЗаполненииВидовДоступа` общего модуля `УправлениеДоступомПереопределяемый`). Для значений, которые не являются значениями доступа, результат проверки будет `Истина` (кроме `Null` и `Неопределено` - результат всегда `Ложь`).
- В разделах ограничения права `Чтение` (`РазрешитьЧтение`, `РазрешитьЧтениеИзменение`) значение будет проверяться в группах доступа с профилем, который предоставляет право `Чтение` списка. В разделах ограничения права `Изменение` (`РазрешитьЧтениеИзменение`, `РазрешитьИзменениеЕслиРазрешеноЧтение`) значение будет проверяться в группах доступа с профилем, который предоставляет право `Изменение` списка. Право `Добавление` на уровне записей приравнено праву `Изменение`, но только для тех групп доступа, профили которых предоставляют право `Добавление` списка.
- Следует иметь в виду особенности использования функции `ЗначениеРазрешено` для проверки значений с типами ссылок справочников `Пользователи`, `Группы пользователей`, `Внешние пользователи`, `Группы внешних пользователей`. Такое использование создает повышенное количество ключей доступа, а также повышенное количество записей в результате расчета прав, так как расчет требуется выполнять с точностью до пользователей, а не с точностью до групп доступа (наборов групп доступа).
- При объединении результатов проверки значений доступа по `ИЛИ` предусмотрено специальное значение `Отключено`, которое можно уточнить `КАК Ложь` (по умолчанию `Истина`). Когда в реквизите обнаружено значение доступа того вида доступа, который отключен по функциональной опции или не указан в профиле группы доступа, это значение доступа считается значением `Отключено`. Следующее ограничение будет работать по оставшемуся виду доступа, когда один из них отключен, но когда оба вида доступа отключены доступ будет разрешен:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
| ЗначениеРазрешено(УчетнаяЗаписьЭлектроннойПочты, Отключено КАК Ложь)
| ИЛИ ЗначениеРазрешено(ОтветственныйПользователь, Отключено КАК Ложь)";
```

Функцию `ЭтоАвторизованныйПользователь` следует использовать только тогда, когда нужно сделать проверку без учета наличия ограничения по виду доступа `Пользователи` или `Внешние пользователи`.

- Функция работает, как отбор вида `Документ.Ответственный = &АвторизованныйПользователь`. Для всех значений, кроме авторизованного пользователя результат проверки будет `Ложь`.
- Следует иметь в виду, что функция создает повышенное количество ключей доступа и записей расчета прав, как и при проверке типов ссылок пользователей в функции `ЗначениеРазрешено`.

Функции `ЧтениеОбъектаРазрешено` и `ИзменениеОбъектаРазрешено` следует использовать, когда нужно сделать зависимость расчета прав на один объект от наличия прав на другой объект. Например, зависимость присоединенного файла от его владельца, строки журнала документов от документа, записи регистра сведений от регистратора и т.п.

- Функция проверяет право на конкретный объект списка по его ссылке. Результат проверки всегда будет `Истина` для объектов, которые не участвуют в ограничении доступа (не указаны в процедуре `ПриЗаполненииСписковСОграничениемДоступа` общего модуля `УправлениеДоступомПереопределяемый`).
- Следует иметь в виду, что ограничения доступа к документу в виде `ЗначениеРазрешено(Организация)` и в виде `ЧтениеОбъектаРазрешено(Организация)` являются разными. В первом случае проверяется разрешенность значения в группах доступа с правами на документ, а во втором случае проверяется, что группа доступа с правами на документ предоставляет право `Чтение` на проверяемую организацию.
- При использовании функций нужно учитывать их влияние на производительность, о чем подробно написано ниже в подразделе Эффективное наследование прав от объекта-владельца.

Функции `ЧтениеСпискаРазрешено` и `ИзменениеСпискаРазрешено` следует использовать, когда нужно сделать зависимость права на один объект от права на список другого объекта. Например, журнал документов имеет несколько типов документов, права на разные типы документов предоставляются пользователям не полностью, а по частям. А значит нужно показывать только те строки журнала, которые соответствуют типам документов, на списки которых у пользователя есть права в ролях.

- Функции проверяют наличие соответствующего права на список в целом, а не на конкретный объект.
- Если для журнала документов решено обойтись без функции `ЧтениеОбъектаРазрешено`, тогда требуется учесть, что не на все документы у пользователя могут быть права и применить функцию `ЧтениеСпискаРазрешено` к полю `Ссылка`.
- Если все документы журнала не используют ограничения, то вместо функции `ЧтениеОбъектаРазрешено` следует использовать функцию `ЧтениеСпискаРазрешено`.
- Следует учитывать, что функции не создают повышенное количество ключей доступа и записей расчета прав, поэтому могут применяться всегда, когда требуются.

Конвертация ограничений прав доступа в стандартный вариант

Для вновь разрабатываемых прикладных решений рекомендуется производительный вариант работы. Однако для плавного перевода пользователей предыдущих версий прикладных решений со стандартного на производительный вариант некоторое время может быть необходимо поддерживать сразу два варианта работы. Схема синхронной разработки подразумевает разработку ограничений доступа для производительного варианта и конвертацию их в стандартный вариант. Обеспечить синхронную разработку поможет отчет `ПроверкаВнедренияБСП.erf`, который проверяет корректность внедрения подсистемы для двух вариантов одновременно.

В стандартном варианте логика ограничения реализуется с помощью параметров одного из четырех шаблонов:

```
#ПоЗначениям(…)
#ПоНаборамЗначений(…)
#ПоЗначениямРасширенный(…)
#ПоЗначениямИНаборамРасширенный(…)
```

> **Примечание**
>
> Подробное описание назначения и параметров шаблонов приведено в комментариях в начале текстов самих шаблонов. Рекомендуется изучить это описание перед прочтением дальнейшего материала. Стандартные шаблоны поставляются в роли `ИзменениеУчастниковГруппДоступа`.

Конвертация типовых случаев логики ограничения в формат параметров шаблонов

Ниже даны примеры конвертации для вариантов подраздела Типовые случаи логики ограничения (см. выше).

- Организация и контрагент в шапке документа:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Контрагент)";
#ПоЗначениям("Документ.ЗаказКлиента", "","",
"Организации", "Организация",
"ГруппыКонтрагентов","Контрагент",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

- Организация в шапке документа, контрагент в табличной части, достаточно одного разрешенного контрагента:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент)";
#ПоЗначениямРасширенный("Документ.ЗаказМатериалов", "", "",
"ЛЕВОЕ СОЕДИНЕНИЕ Документ.ЗаказМатериалов.Поставщики КАК Т1
ПО Т1.Ссылка = Т.Ссылка",
"",
"Организации", "Т.Организация","И",
"ГруппыКонтрагентов", "Т1.Контрагент", "",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Организация в шапке документа, контрагент в табличной части, достаточно одного разрешенного контрагента (если табличная часть пустая, тогда доступ по контрагенту разрешен):

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент, Null КАК Истина)";
#ПоЗначениямРасширенный("Документ.ЗаказМатериалов", "", "",
"ЛЕВОЕ СОЕДИНЕНИЕ Документ.ЗаказМатериалов.Поставщики КАК Т1
ПО Т1.Ссылка = Т.Ссылка",
"",
"Организации", "Т.Организация","И(",
"ГруппыКонтрагентов", "Т1.Контрагент", "ИЛИ",
"Условие", "Т1.Контрагент ЕСТЬ NULL",")",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Организация в шапке документа, контрагент в табличной части, и требуется, чтобы все контрагенты были разрешены (если табличная часть пустая, тогда доступ по контрагенту запрещен):

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Организация)
|И ДляВсехСтрок(ЗначениеРазрешено(Поставщики.Контрагент))";
#ПоЗначениямРасширенный("Документ.ЗаказМатериалов", "", "",
"",
"",
"Организации", "Т.Организация","И
НЕ ЛОЖЬ В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ЛОЖЬ
  ИЗ Документ.ЗаказМатериалов.Поставщики КАК Т1
  ГДЕ Т.Ссылка = Т1.Ссылка И НЕ (",
"ГруппыКонтрагентов", "Т1.Контрагент", "))И
ИСТИНА В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ИСТИНА
  ИЗ Документ.ЗаказМатериалов.Поставщики)",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Организация и контрагент в табличной части, при этом достаточно, чтобы любая из пар организации с контрагентом была разрешена:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Поставщики.Организация)
|И ЗначениеРазрешено(Поставщики.Контрагент)";
#ПоЗначениямРасширенный("Документ.ПланированиеЗакупок", "", "",
"ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПланированиеЗакупок.Материалы КАК Т1
ПО Т1.Ссылка = Т.Ссылка",
"",
"Организации", "Т1.Организация","И",
"ГруппыКонтрагентов", "Т1.Контрагент", "",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Организация и контрагент в табличной части, при этом требуется, чтобы все пары организации с контрагентом были разрешены:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ДляВсехСтрок(
|      ЗначениеРазрешено(Поставщики.Организация)
|    И ЗначениеРазрешено(Поставщики.Контрагент))";
#ПоЗначениямРасширенный("Документ.ПланированиеЗакупок", "", "",
"",
"НЕ ЛОЖЬ В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ЛОЖЬ
  ИЗ Документ.ЗаказМатериалов.Материалы КАК Т1
  ГДЕ Т.Ссылка = Т1.Ссылка И НЕ (",
"Организации", "Т1.Организация","И",
"ГруппыКонтрагентов", "Т1.Контрагент", "))И
ИСТИНА В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ИСТИНА
  ИЗ Документ.ЗаказМатериалов.Материалы)",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Организация и контрагент в табличной части, при этом требуется, чтобы одна из организаций и один из контрагентов были разрешены:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ДляОднойИзСтрок(ЗначениеРазрешено(Поставщики.Организация))
|И ДляОднойИзСтрок(ЗначениеРазрешено(Поставщики.Контрагент))";
#ПоЗначениямРасширенный("Документ.ПланированиеЗакупок", "", "",
"",
"ИСТИНА В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ИСТИНА
  ИЗ Документ.ЗаказМатериалов.Материалы КАК Т1
  ГДЕ Т.Ссылка = Т1.Ссылка И",
"Организации", "Т1.Организация",") И
ИСТИНА В (
  ВЫБРАТЬ ПЕРВЫЕ 1 ИСТИНА
  ИЗ Документ.ЗаказМатериалов.Материалы КАК Т1
  ГДЕ Т.Ссылка = Т1.Ссылка И",
"ГруппыКонтрагентов", "Т1.Контрагент", ")",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
```

- Отправитель – измерение составного типа, при этом требуется проверять только ссылки `Справочник.Склады`:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|  ЗначениеРазрешено(Отправитель ТОЛЬКО Справочник.Склады)";
#ПоЗначениямРасширенный("Документ.ПеремещениеТоваров", "", "",
"",
"ВЫБОР КОГДА ТИПЗНАЧЕНИЯ(Т.Отправитель) = ТИП(Справочник.Склады) ТОГДА",
"Отправители", "Т.Отправитель","ИНАЧЕ ИСТИНА КОНЕЦ",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","",
```

- Организация в шапке (Владелец), головная организация в справочнике организаций, при этом требуется, чтобы изменение было разрешено, когда разрешена организация в шапке, а чтение было разрешено:
  - когда разрешена организация в шапке (Владелец),
  - когда разрешена непустая головная организация организации в шапке,
  - когда разрешена организация, у которой организация в шапке (Владелец) является головной.

```
Ограничение.Текст =
"ПрисоединитьДополнительныеТаблицы
|ЭтотСписок КАК ЭтотСписок
|
|ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК Владельцы
|    ПО Владельцы.Ссылка = ЭтотСписок.Владелец
|
|ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК ОбособленныеПодразделения
|    ПО ОбособленныеПодразделения.ГоловнаяОрганизация = Владельцы.Ссылка
|;
|РазрешитьЧтение
|ГДЕ
|    ЗначениеРазрешено(Владелец)
| ИЛИ ЗначениеРазрешено(Владельцы.ГоловнаяОрганизация, ПустаяСсылка КАК Ложь)
| ИЛИ ЗначениеРазрешено(ОбособленныеПодразделения.Ссылка)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ЗначениеРазрешено(Владелец)";
#ПоЗначениямРасширенный("Справочник.ПодразделенияОрганизаций", "Чтение", "",
"ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК Т1
ПО Т1.Ссылка = Т.Владелец
ЛЕВОЕ СОЕДИНЕНИЕ Справочник.Организации КАК Т2
ПО Т2.ГоловнаяОрганизация = Т1.Ссылка",
"",
"Организации", "Т.Владелец","ИЛИ",
"Организации", "Т1.ГоловнаяОрганизация", "И",
"Условие", "Т1.ГоловнаяОрганизация <> ЗНАЧЕНИЕ(Справочник.Организации.ПустаяСсылка)", "ИЛИ",
"Организации", "Т2.Ссылка","",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","")
#ПоЗначениям("Справочник.ПодразделенияОрганизаций", "Изменение", "",
"Организации", "Владелец",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

- Перенос прав от объекта-владельца (например, на присоединенные файлы).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(ВладелецФайла)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(ВладелецФайла)";
```

  - когда мало типов в реквизите `ВладелецФайла`:

```
#ПоЗначениям("Справочник.ЗаказКлиентаПрисоединенныеФайлы", "","",
"Организации", "ВладелецФайла.Организация",
"ГруппыКонтрагентов"," ВладелецФайла.Контрагент",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

  - когда много типов в реквизите `ВладелецФайла` (требуется запись наборов значений доступа для всех владельцев):

```
#ПоЗначениямИНаборамРасширенный("Справочник.Файлы","","",
"",
"",
"Объект","Т.ВладелецФайла","",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","
```

- Ограничение по чтению владельца (например, журнал документов).

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|    ЧтениеОбъектаРазрешено(Ссылка)";
```

  - когда в графы можно перенести все поля для условия ограничения:

```
#ПоЗначениям("ЖурналДокументов.Заказы", "","",
"Организации", "Организация",
"ГруппыКонтрагентов"," Контрагент",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

  - в противном случае и когда мало типов в реквизите Ссылка:

```
#ПоЗначениям("ЖурналДокументов.Заказы", "","",
"Организации", "Ссылка.Организация",
"ГруппыКонтрагентов","Ссылка.Контрагент",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

   - в противном случае и когда много типов в реквизите Ссылка (требуется запись наборов значений доступа для всех владельцев):

```
#ПоЗначениямИНаборамРасширенный("ЖурналДокументов.Заказы","","",
"",
"",
"Объект","Т.Ссылка","",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","
```

- Проверка настроек прав Чтение и Изменение на объекты (например, настройки прав на элементы справочника `ПапкиФайлов`).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(Ссылка)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(Ссылка)";
#ПоЗначениям("Справочник.ПапкиФайлов", "", "",
"НастройкиПрав", "Ссылка",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

- Проверка настроек прав Чтение и Изменение на дополнительные объекты (например, настройки прав на справочник `Файлы`, сделанные по элементам справочника `ПапкиФайлов`).

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(ВладелецФайла)
|;
|РазрешитьИзменениеЕслиРазрешеноЧтение
|ГДЕ
|    ИзменениеОбъектаРазрешено(ВладелецФайла)";
```

   - когда `ВладелецФайла` только типа справочника `ПапкиФайлов`:

```
#ПоЗначениям("Справочник.Файлы", "", "",
"НастройкиПрав", "ВладелецФайла",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

   - когда `ВладелецФайла` составного типа (требуется запись наборов значений доступа):

```
#ПоЗначениямИНаборамРасширенный("Справочник.Файлы", "", "",
"",
"",
"Объект","Т.ВладелецФайла","",
"","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","
```

- Учетная запись и контрагент в шапке, при этом достаточно, чтобы было разрешение для одного из них. При этом если отключено ограничение по учетным записям, должно остаться ограничение по контрагентам, а если отключено ограничения по контрагентам - остается ограничение по учетным записям. Если отключены оба ограничения, то доступ должен быть разрешен.

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
| ЗначениеРазрешено(УчетнаяЗапись, Отключено КАК Ложь)
| ИЛИ ЗначениеРазрешено(Контрагент, Отключено КАК Ложь)";
#ПоНаборамЗначений("Документ.Письмо","","РасширенноеИЛИ","")
```

Требуется запись наборов значений доступа, в модуле объекта документа Письмо:

```
Процедура ЗаполнитьНаборыЗначенийДоступа(Таблица) Экспорт

    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 1;
    Строка.ЗначениеДоступа = УчетнаяЗапись;
    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 2;
    Строка.ЗначениеДоступа = Контрагент;

КонецПроцедуры
```

Особенности и назначение шаблонов ограничения доступа

Наличие сразу нескольких шаблонов объясняется необходимостью решать различные задачи. Так, шаблоны `#ПоЗначениям` и `#ПоНаборамЗначений` позволяют решить большую часть простых задач.

В шаблонах в качестве параметров имен видов доступа также используются специальные функции `Условие`, `Объект`, `НастройкиПрав`, `ПравоЧтения`, `ПравоИзменения`. Функция `Условие` доступна во всех шаблонах. Функция `Объект` доступна только в шаблоне `ПоЗначениямИНаборамРасширенный`. Функции `НастройкиПрав`, `ПравоЧтения` и `ПравоИзменения` доступны в шаблонах `ПоЗначениям`, `ПоЗначениямРасширенный`, `ПоЗначениямИНаборамРасширенный`. Эти функции тоже называются видами доступа, хотя их нет нигде, кроме указанных стандартных шаблонов.

Специальная функция `Условие` в параметре имени вида доступа означает, что в параметре имени реквизита находится произвольное условие на языке запросов.

Специальные функции `Объект` и `НастройкиПрав` в ограничении права `Чтение`, работают аналогично функции `ЧтениеОбъектаРазрешено` (производительный вариант), а в ограничении прав `Изменение` и `Добавление` аналогично функции `ИзменениеОбъектаРазрешено` (производительный вариант).

Специальная функция `НастройкиПрав` в параметре имени вида доступа означает, что в параметре имени реквизита указан реквизит с типом из определяемого типа `ВладелецНастроекПрав`. При этом будет осуществляться только проверка настроек прав, а если тип другой, тогда результат проверки будет Ложь.

Для шаблонов `#ПоЗначениямИНаборамРасширенный` (только для вида доступа `Объект`) и `#ПоНаборамЗначений` можно использовать модификатор `РасширенноеИЛИ`, который меняет расчет ограничений с несколькими проверками, результаты которых объединяются по `ИЛИ`. Модификатор работает аналогично уточнению `Отключено КАК Ложь` функции `ЗначениеРазрешено` (производительный вариант). Например, электронное письмо доступно, если разрешена `Учетная запись` или `Организация`. Без модификатора, если ограничение по учетным записям или организациям не используется, все письма будут разрешены. Случаи отключения: выключена константа использования вида доступа, нет вида доступа в профиле, отключено ограничение доступа на уровне записей в целом.

Шаблоны с модификатором `РасширенноеИЛИ` работают медленнее, поэтому не следует использовать модификатор там, где нет логики ограничения по `ИЛИ`. Если в ограничении используется вид доступа `Объект`, а типы объектов и их логика ограничения заранее неизвестны, модификатор нужно указать, когда требуется расширенный расчет `ИЛИ`.

Шаблоны поддерживают список видов доступа в параметре имени вида доступа (в алфавитном порядке через запятую). В этот список не входят специальные виды доступа (`Условие`, `Объект`, `НастройкиПрав`, `ПравоЧтения`, `ПравоИзменения`). Если проверяется реквизит составного типа, например, реквизит `Владелец` типов `Организация` и `Контрагент`, то можно сделать более простое и эффективное ограничение (без анализа типов и двух проверок вместо одной). Например:

```
#ПоЗначениям("Справочник.БанковскиеСчета", "", "",
"ГруппыКонтрагентов,Организации","Владелец",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

Шаблон «#ПоЗначениям»

Шаблон `#ПоЗначениям` позволяет реализовать проверку разрешения значений доступа в полях таблицы (указывать виды доступа нужно, чтобы текст ограничения автоматически оптимизировался, когда вид доступа не используется). Например, текст ограничения доступа вида:

```
#ПоЗначениям("Документ.ОприходованиеТоваров", "", "",
"Организации", "Организация",
"Склады", "Склад",
"","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","", "","")
```

ограничивает чтение таким образом, что пользователь увидит в списке документов **Оприходование товаров** только те документы, у которых в реквизите `Организация` есть значение, разрешенное пользователю по виду доступа **Организации**, и в реквизите `Склад` есть значение, разрешенное по виду доступа **Склады**. Причем значения должны быть разрешены в группе доступа пользователей одновременно, а профиль групп доступа пользователей должен включать роль с правом чтения документов **Оприходование товаров**.

В шаблоне можно выполнить максимум 16 проверок.

Этот и другие шаблоны поддерживают передачу условия на языке запросов. Для этого вместо имени вида доступа нужно указать имя **Условие**, а вместо имени поля – текст на языке запросов. Например:

```
#ПоЗначениям("Документ.ОприходованиеТоваров", "", "",
"Организации", "Организация",
"Склады", "Склад",
"Условие", "Т.ХозяйственнаяОперация = Значение(Перечисление.ХозяйвственныеОперации.Инветаризация)", "","", "","", "","", "","", "","", "","", "","", "
```

В этом примере добавляется условие: документ доступен только при хозяйственной операции **Инвентаризация**. Из этого следует, что при другой хозяйственной операции, например **Прочее**, документ не будет доступен ни одному пользователю, кроме пользователей с ролью **Полные права**.

Шаблон «#ПоНаборамЗначений»

В первую очередь шаблон `#ПоНаборамЗначений` предназначен для ограничения чтения журналов документов, у которых документы имеют различные ограничения чтения, например по составу реквизитов. Обычно для журнала приходится делать анализ типа документа, а затем применять те же проверки, что и для документа соответствующего типа, т. е. повторять логику ограничений всех документов в каждом журнале, в

который входят документы. Как правило, этот способ очень «дорогой» в плане производительности и трудоемкий с точки зрения разработки и сопровождения. Поэтому целесообразно предварительно записывать значения доступа объектных элементов данных, необходимые для работы логики ограничения доступа, в специальный регистр – `НаборыЗначенийДоступа`. Такие данные называются наборами значений доступа.

Например, если для документа **Оприходование товаров** в регистре `НаборыЗначенийДоступа` имеются записи:

```
<Документ1>, , "Организации", "Организация1"
<Документ1>, , "Склады", "Склад1";
```

то для их использования текст ограничения права чтения журнала, содержащего этот документ, должен быть таким:

```
#ПоНаборамЗначений("ЖурналДокументов.СкладскиеДокументы", "", "", "Ссылка"),
```

где `Ссылка` – имя стандартного реквизита журнала.

В этом случае логика работы шаблона `#ПоНаборамЗначений` такова: по переданному значению стандартного реквизита `Ссылка` строки журнала в регистре **Наборы значений доступа** будут найдены все строки, содержащие **Значение доступа**, и выполнена проверка, что все значения доступа одновременно разрешены пользователю в одной из его групп доступа.

Помимо журналов документов шаблон `#ПоНаборамЗначений` может применяться для реализации логики ограничения доступа и к другим объектам метаданных. При этом использование заранее подготовленных наборов значений позволяет избежать тех ограничений, которые накладываются языком запросом. Применение шаблона `#ПоНаборамЗначений` требует дополнительной настройки объекта метаданных. К объекту метаданных необходимо добавить табличную часть **Наборы значений доступа** с реквизитами, совпадающими с измерениями и ресурсами регистра сведений **Наборы значений доступа**. В тексте ограничения доступа в качестве четвертого параметра шаблона `#ПоНаборамЗначений` указывается пустая строка. Также требуется указать типы этих объектов в определяемом типе `ВладелецСОграничениемПоНаборамЗначенийДоступаОбъект` или `ВладелецСОграничениемПоНаборамЗначенийДоступаДокумент`.

Шаблон «#ПоЗначениямРасширенный»

В некоторых случаях возможностей шаблонов `#ПоЗначениям` и `#ПоНаборамЗначений` бывает недостаточно. Например, требуется ограничить документ **Перемещение товаров** так, чтобы чтение документа было доступно пользователю исходя из разрешенности `Организации` и `СкладаПолучателя` или `СкладаОтправителя`, а запись документа была доступна только при разрешенности всех трех: `Организации`, `СкладаПолучателя` и `СкладаОтправителя`. В этом случае право `Запись` можно ограничить, применив шаблон `#ПоЗначениям`, а право `Чтение` – нет.

Для ограничений, предполагающих логику с `ИЛИ`, предусмотрены расширенные шаблоны, в частности `#ПоЗначениямРасширенный`. Например, ограничение для права `Чтение` выглядит следующим образом:

```
#ПоЗначениямРасширенный("Документ.ПеремещениеТоваров", "", "",
"",
"",
"Организации", "Т.Организация", "И(",
"Склады", "Т.СкладОтправитель", "ИЛИ",
"Склады", "Т.СкладПолучатель", ")", …)
```

Требуется явно указывать псевдоним текущей таблицы «Т.» перед полем.

Кроме реализации логики `ИЛИ` расширенные шаблоны могут использоваться для проверки реквизитов табличных частей – через присоединяемые таблицы. Для того чтобы к основной таблице присоединить табличную часть, в четвертый параметр шаблона нужно передать текст на языке запросов, например:

```
"Внутреннее Соединение Документ.ПеремещениеТоваров.Номенклатура
КАК Т2 по Т.Ссылка = Т2.Ссылка",
```

Для шаблона основная таблица расширяется полями табличной части, а за счет явного применения псевдонимов их можно указывать подобно полям основной таблицы. Т. е. в шаблон можно добавить проверку реквизита табличной части, например:

```
"ГруппыНоменклатуры", "Т2.Номенклатура", ""
```

В результате получится такое ограничение:

```
#ПоЗначениямРасширенный("Документ.ПеремещениеТоваров", "", "",
"Внутреннее Соединение Документ.ПеремещениеТоваров.Номенклатура КАК Т2 по Т.Ссылка = Т2.Ссылка",
"",
"Организации", "Т.Организация", "И(",
"Склады", "Т.СкладОтправитель", "ИЛИ",
"Склады", "Т.СкладПолучатель", ")И",
"ГруппыНоменклатуры", "Т2.Номенклатура", "", …)
```

Когда основная таблица расширяется полями присоединенной таблицы, происходит умножение строк. Например, табличная часть содержит 2 строки с разной номенклатурой. В этом случае документ будет доступен, если выполнились условия проверки основной таблицы («шапки» документа) и доступна хотя бы одна из двух номенклатур табличной части.

Шаблон «#ПоЗначениямИНаборамРасширенный»

Для решения задачи ограничения доступа к объектам метаданных в зависимости от ограничений доступа к другим объектам может использоваться шаблон `#ПоЗначениямИНаборамРасширенный`.

Например, файл (элемент справочника **Файлы**), прикрепленный к документу **Поступление товаров и услуг**, должен быть доступен пользователю при тех же условиях, что и сам документ.

Такая задача может быть решена «повтором» логики ограничения документа **Поступление товаров и услуг** в ограничении справочника **Файлы**. Однако на практике это приводит к необходимости синхронно изменять логику ограничения документов и файлов. Кроме того, поскольку элементы справочника **Файлы** могут «прикрепляться» к произвольному числу объектов метаданных (например, к папкам файлов, документам **Заказ**, **Расходная накладная** и т. п.), задача реализации логики ограничения доступа к файлам сравнима по сложности с ограничением доступа к журналу документов, который содержит все виды документов конфигурации.

В некоторых случаях, когда число владельцев небольшое (не более 10), можно повторять логику ограничения владельца и при этом иметь приемлемые показатели производительности, особенно если логика ограничения владельцев совпадает. При повторении логики ограничения нужно учесть, что проверить значения доступа владельцев недостаточно, а нужно еще учесть наличие прав на таблицу владельца, иначе будут доступны файлы владельцев, которые сами недоступны по правам (по составу ролей). Для этой цели нужно использовать специальные виды доступа `ПравоЧтения` и `ПравоИзменения`, которые позволяют проверить соответственно наличие прав `Чтение` и `Изменение` на таблицу владельца, например:

```
#ПоЗначениям("Справочник.Файлы", "Чтение", "",
"ПравоЧтение", "Т.ВладелецФайла",
"Организации", "Т.ВладелецФайла.Организация", …).
```

Когда владельцев много, этот способ не подходит. Для такого случая подсистема **Управление доступом** предоставляет специальный вид доступа по объекту-владельцу. Для использования этого вида доступа в шаблоне следует указывать в качестве имени вида доступа **Объект**. Вид доступа **Объект** предполагает, что в поле находится значение, которое является ссылкой на объект данных (например, документ). А в регистре **Наборы значений доступа**, как и для шаблона `#ПоНаборамЗначений`, записаны наборы значений для этого объекта, которые следует проверить. Исключение составляют владельцы настроек прав – для них наборы значений доступа записывать не требуется, но процедура заполнения может понадобиться, если они используются для заполнения наборов значений доступа зависимых объектов.

Так, используя шаблон `#ПоЗначениямИНаборамРасширенный`, можно проверить не одно значение доступа, а набор значений, подготовленный при записи документа, ссылка на который находится в реквизите, например, `Владелец` элемента справочника `Файлы`.

```
#ПоЗначениямИНаборамРасширенный("Справочник.Файлы", "", "",
"",
"",
"Объект","Т.ВладелецФайла","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", "","","", ""
```

Наборы значений элементов справочника **Файлы** требуют связанного обновления при изменении, например, наборов значений доступа документа – владельца файлов **Поступление товаров и услуг**.

При использовании вида доступа **Объект** кроме проверки наборов выполняется еще проверка прав на объект (то есть добавляется проверка, которая используется в видах доступа `ПравоЧтения` и `ПравоИзменения`). В ограничении права чтения таблицы выполняется проверка права чтения объекта и наборов ограничения чтения объекта, а в ограничении прав добавления, изменения, удаления таблицы выполняется проверка права изменения объекта и наборов ограничения изменения объекта.

Права объекта, которые проверяются в ограничении по виду доступа **Объект**, можно переопределить в процедуре `ПриЗаполненииЗависимостейПравДоступа` модуля `УправлениеДоступомПереопределяемый`.

Структура регистра **Наборы значений доступа** поддерживает логику `ИЛИ` для значений доступа и различные наборы по правам доступа. Для реализации логики `ИЛИ` добавлено измерение **Номер набора** типа `Число`, а для наборов по правам добавлены два ресурса: `Чтение`, `Изменение` – типа `Булево`.

Сложную логику `ИЛИ` записать в регистр «напрямую» не получится. Номер набора позволяет сделать для одного ссылочного элемента данных (например, документа) несколько наборов значений, по которым проверяется доступность документа. Причем достаточно, чтобы значения хотя бы одного набора значений были доступны одновременно. Точнее, результаты проверки значений доступа по видам доступа в пределах набора с одним номером объединяются по `И`, а результаты проверки целых наборов объединяются по `ИЛИ`. Например, для документа **Перемещение товаров** регистр **Наборы значений доступа** будет содержать записи для логики права `Чтение`:

```
<Документ2>, 0, "Организации", "Организация1"
<Документ2>, 0, "Склады", "Склад1";
<Документ2>, 1, "Организации", "Организация1"
<Документ2>, 1, "Склады", "Склад2";
```

Видно, что, хотя значений доступа три: `Организация1`, `Склад1`, `Склад2`, – записей в регистре четыре. Это связано с раскрытием скобок в логическом выражении **О1** и (**С1** или **С2**), что приводит к логическому выражению суммы произведений **О1** и **С1** или **О1** и **С2**.

Действие раскрытия скобок первичного логического выражения, объединяющего результаты проверки значений доступа, необходимо для заполнения наборов записей регистра сведений **Наборы значений доступа**.

Записи для логики права `Изменение`:

```
<Документ2>, 0, "Организации", "Организация1"
<Документ2>, 0, "Склады", "Склад1";
<Документ2>, 0, "Склады", "Склад2";
```

Пример процедуры заполнения приведен ниже.

Шаблон `#ПоЗначениямИНаборамРасширенный` отличается от шаблона `#ПоЗначениямРасширенный` только поддержкой специального вида доступа **Объект**.

Разработка процедур «Заполнить наборы значений доступа»

При использовании стандартных шаблонов `#ПоНаборамЗначений` или `#ПоЗначениямИНаборамЗначений` с видом доступа **Объект** потребуется записать в регистр **Наборы значений доступа** сведения, требующиеся для их работы.

Запись наборов значений доступа выполняется при записи объектов – например, документов. Перед записью вызывается пользовательская процедура заполнения наборов значений доступа.

Процедура заполнения помещается в модуль объекта, и его тип включается в подписку на события **Записать наборы значений доступа** и в состав типов измерения `Объект` регистра сведений **Наборы значений доступа**.

При заполнении можно использовать только следующие свойства таблицы:

- **Номер набора** (если набор один, то заполнять не нужно);
- **Вид доступа** (только для специальных видов доступа `ПравоЧтения`, `ПравоИзменения`);
- **Значение доступа** (назначить явно);
- **Чтение**, **Изменение** (если набор для всех прав, то заполнять не нужно).

Количество заполненных записей должно быть больше нуля, иначе будет вызвано исключение. Это важно, т. к. процедура заполнения наборов значений доступа после включения режима **Ограничивать доступ на уровне записей** вызывается из регламентного задания **Заполнение данных для ограничения доступа** для тех объектов, которые имеют пустой набор записей в регистре **Наборы значений доступа**. При выключении режима **Ограничивать доступ на уровне записей** производится запись пустого набора. Следовательно, при включении режима выполнится заполнение очищенных наборов (или полное заполнение). Регламентное задание вызывается многократно и отрабатывает порцию заполнения. Очищение наборов происходит «по ходу» перезаписи объектов.

Технология порционного заполнения/очистки использует пустой набор записей как флажок состояния. Когда нужно записать заведомо «запрещенный набор», записывается не пустой набор, а следующий:

```
Строка = Таблица.Добавить();
Строка.ЗначениеДоступа = Перечисления.ДополнительныеЗначенияДоступа.ДоступЗапрещен;
```

Когда нужно записать заведомо «разрешенный набор», записывается не пустой набор, а следующий:

```
Строка = Таблица.Добавить();
Строка.ЗначениеДоступа = Перечисления.ДополнительныеЗначенияДоступа.ДоступРазрешен;
```

Пример процедуры:

`ЗаполнитьНаборыЗначенийДоступа` в модуле объекта – тип объекта включается в определяемый тип `ВладелецНаборовЗначенийДоступаОбъект`.

```
Процедура ЗаполнитьНаборыЗначенийДоступа(Таблица) Экспорт

    Строка = Таблица.Добавить();
    Строка.Изменение       = Истина;
    Строка.ЗначениеДоступа = Организация1;
    Строка = Таблица.Добавить();
    Строка.ЗначениеДоступа = Склад1;
    Строка = Таблица.Добавить();
    Строка.ЗначениеДоступа = Склад2;
    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 1;
    Строка.Чтение          = Истина;
    Строка.ЗначениеДоступа = Организация1;
    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 1;
    Строка.ЗначениеДоступа = Склад1;
    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 2;
    Строка.Чтение          = Истина;
    Строка.ЗначениеДоступа = Организация1;
    Строка = Таблица.Добавить();
    Строка.НомерНабора     = 2;
    Строка.ЗначениеДоступа = Склад2;

КонецПроцедуры
```

Если в процедуре заполнения не задан номер набора, то считается, что добавлен один набор с номером 1. Если в пределах набора с одним номером не указано ни одно из прав `Чтение` или `Изменение`, то считается, что набор относится ко всем правам.

В наборе допускается только одна запись с одним и тем же значением доступа, дубли строк и наборов удаляются автоматически в процедуре `ДобавитьНаборыЗначенийДоступа`, которая вызывается для этих целей перед записью наборов значений доступа в регистр сведений `НаборыЗначенийДоступа` со сжатием наборов (лишние наборы удаляются по правилам упрощения логических функций).

> **Внимание!**
>
> Для того чтобы изменения вступили в силу сразу при разработке и отладке конфигурации, требуется обновить вспомогательные данные. См. раздел Обновление вспомогательных данных во время разработки.

Зависимые наборы значений доступа

В случае, когда наборы значений доступа нужно сформировать с учетом доступа к другому объекту (к примеру, доступ к бизнес-процессу `_ДемоЗаданиеСРолевойАдресацией` определяется по доступу к его объекту-основанию, например по элементу справочника **Файлы**), появляется необходимость включать в состав одних наборов другие наборы.

Для реализации включения одних наборов в другие разработана процедура `ДобавитьНаборыЗначенийДоступа` общего модуля `УправлениеДоступом`, которая добавляет в таблицу-приемник `Приемник` наборы из таблицы-источника `Источник` либо логическим сложением, либо логическим умножением. Например, наборы задачи:

```
Процедура ЗаполнитьНаборыЗначенийДоступа(Таблица) Экспорт

    Строка = Таблица.Добавить();
    Строка.ЗначениеДоступа = Исполнитель;
    // Создание пустой таблицы.
    ТаблицаБизнесПроцесса = УправлениеДоступом.ТаблицаНаборыЗначенийДоступа();

    // Заполнение наборов бизнес-процесса.
    УправлениеДоступом.ЗаполнитьНаборыЗначенийДоступа(БизнесПроцесс, ТаблицаБизнесПроцесса);
    // Логическое сложение наборов в таблицах (по "ИЛИ").
    УправлениеДоступом.ДобавитьНаборыЗначенийДоступа(Таблица, ТаблицаБизнесПроцесса);

КонецПроцедуры
```

Следует разработать процедуры `ЗаполнитьНаборыЗначенийДоступа` для всех объектов, которые могут быть объектом-основанием или задействованы при формировании наборов значений доступа.

Могут существовать объекты, у которых наборы значений доступа не записываются в регистр сведений `НаборыЗначенийДоступа`, а используются только при формировании других записываемых наборов значений доступа. Типы таких объектов нужно включить в определяемый тип `ВладелецВнешнихЗначенийВНаборахЗначенийДоступаОбъект`.

Для реализации зависимостей необходимо разработать логику вызова перезаписи наборов значений доступа зависимых объектов в обработчике `ПриИзмененииНаборовЗначенийДоступа` общего модуля `УправлениеДоступомПереопределяемый`, например:

```
Процедура ПриИзмененииНаборовЗначенийДоступа(Знач Ссылка, ССылкиНаЗависимыеОбъекты) Экспорт

    ПолноеИмя = Ссылка.Метаданные().ПолноеИмя();

    Если ПолноеИмя = "Задача.ЗадачаИсполнителя" Тогда
        // Зависимые типы объектов:
        //  БизнесПроцесс.Задание
        Запрос = Новый Запрос(
        "ВЫБРАТЬ
        |    Задание.Ссылка
        |ИЗ
        |    БизнесПроцесс.Задание КАК Задание
        |ГДЕ
        |    Задание.Предмет = &Предмет");
        Запрос.УстановитьПараметр("Предмет", Ссылка);
        ССылкиНаЗависимыеОбъекты = Запрос.Выполнить().Выгрузить().ВыгрузитьКолонку("Ссылка");

    ИначеЕсли ПолноеИмя = "БизнесПроцесс.Задание" Тогда
        // Зависимые типы объектов:
        //  БизнесПроцесс.Задание
        // через Задача.ЗадачаИсполнителя в основании.
        Запрос = Новый Запрос(
        "ВЫБРАТЬ РАЗЛИЧНЫЕ
        |    Задание.Ссылка
        |ИЗ
        |    БизнесПроцесс.Задание КАК Задание
        |        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Задача.ЗадачаИсполнителя КАК ЗадачаИсполнителя
        |        ПО Задание.Предмет = ЗадачаИсполнителя.Ссылка
        |            И (ЗадачаИсполнителя.БизнесПроцесс = &БизнесПроцесс)");
        Запрос.УстановитьПараметр("БизнесПроцесс", Ссылка);
        ССылкиНаЗависимыеОбъекты = Запрос.Выполнить().Выгрузить().ВыгрузитьКолонку("Ссылка");

    КонецЕсли;
КонецПроцедуры
```

Специальные имена видов доступа `ПравоЧтения`, `ПравоИзменения` требуются, когда нужно разработать ограничение с зависимостью по правам «подчиненного» объекта от «ведущего» объекта. В шаблонах ОДД в проверку по виду доступа **Объект** встроена логика проверки зависимых прав, как и в этих видах доступа. В связи с этим сочетать данные виды доступа с видом доступа **Объект** не требуется. Также не требуется использовать эти виды доступа при заполнении зависимых наборов значений доступа. Для этих целей предусмотрен третий параметр в процедуре `ЗаполнитьНаборыЗначенийДоступа`, при указании которого соответствующие проверки добавляются автоматически.

> **Внимание!**
>
> Для того чтобы изменения вступили в силу сразу при разработке и отладке конфигурации, требуется обновить вспомогательные данные. См. раздел Обновление вспомогательных данных во время разработки.

Описание видов доступа, используемых в ограничениях объектов для отчета «Права доступа»

В стандартном варианте ограничения доступа, для использования отчета **Права доступа** требуется задать список видов доступа, используемых в ограничении прав объектов метаданных. Список задается в переопределяемом модуле в виде многострочной строки формата `<Полное имя таблиц>.<Право>.<Вид доступа>`. Для автоматического формирования кода процедуры следует применить инструменты разработчика. Пример заполнения процедуры:

```
Процедура ПриЗаполненииВидовОграниченийПравОбъектовМетаданных(Описание) Экспорт

    Описание =
    "
    |БизнесПроцесс.Задание.Добавление.Пользователи
    |БизнесПроцесс.Задание.Изменение.Пользователи
    |БизнесПроцесс.Задание.Чтение.Пользователи
    |…
    |…
    |…
    |Справочник.ПапкиФайлов.Изменение._ДемоПапкиФайлов
    |Справочник.ПапкиФайлов.Чтение._ДемоПапкиФайлов
    |Справочник.УчетныеЗаписиЭлектроннойПочты.Чтение.УчетныеЗаписиЭлектроннойПочты
    |";

КонецПроцедуры
```

Настройка свойств объектов метаданных

Свойство объекта метаданных	Настройка
`ОпределяемыйТип.ЗначениеДоступа`	• Обязательные типы: • `СправочникСсылка.Пользователи`, • `СправочникСсылка.ГруппыПользователей`, • `СправочникСсылка.ВнешниеПользователи`, • `СправочникСсылка.ГруппыВнешнихПользователей`. • Типы значений доступа, например `СправочникСсылка.Организации`. • Типы групп значений доступа, например `СправочникСсылка._ДемоГруппыДоступаКонтрагентов`
`ОпределяемыйТип.ЗначениеДоступаОбъект`	• Типы значений доступа, например `СправочникСсылка.Организации`. • Типы значений доступа, для которых указан тип групп значений (например, группы доступа контрагентов `СправочникОбъект._ДемоКонтрагенты`). Если типов нет, нужно указать тип `СправочникОбъект.ИдентификаторыОбъектовМетаданных` (для корректного заполнения свойств подписок на событие `ОбновитьГруппыЗначенийДоступаПередЗаписью`, `ОбновитьГруппыЗначенийДоступаПриЗаписи`).
`ОпределяемыйТип.ВладелецНастроекПрав`	• Типы владельцев настроек прав, например `СправочникСсылка.ПапкиФайлов`.
`ОпределяемыйТип.ВладелецНастроекПравОбъект`	• Типы владельцев настроек прав, например `СправочникОбъект.ПапкиФайлов`. Если типов нет, нужно указать тип `СправочникОбъект.ИдентификаторыОбъектовМетаданных` (для корректного заполнения свойств подписки на событие `ОбновитьГруппыВладельцевНастроекПрав`).
`ОпределяемыйТип.ВладелецЗначенийКлючейДоступа`	Все ссылочные объекты с ограничением доступа, а также объекты без ограничения, которые записывают ключи доступа для других объектов, например журналов документов. Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`ОпределяемыйТип.ВладелецЗначенийКлючейДоступаОбъект`	Все ссылочные объекты (кроме документов) из определяемого типа `ВладелецЗначенийКлючейДоступа`, а также объекты, которые участвуют в ограничениях доступа других объектов (обращение «через точку» или путем соединения). Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`ОпределяемыйТип.ВладелецЗначенийКлючейДоступаДокумент`	Все документы из определяемого типа `ВладелецЗначенийКлючейДоступа`, а также документы, которые участвуют в ограничениях доступа других объектов (обращение «через точку» или путем соединения). Если документов нет, нужно указать тип `СправочникОбъект.ИдентификаторыОбъектовМетаданных` (для корректного заполнения свойств подписок на событие `ПроверитьДоступПередЗаписьюДокумента`, `ПроверитьДоступПриЗаписиДокумента`, `ПроверитьДоступПередУдалениемДокумента`). Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`ОпределяемыйТип.ВладелецЗначенийКлючейДоступаНаборЗаписей`	Все регистры с ограничением (кроме регистров расчета), а также регистры, которые участвуют в ограничениях доступа других объектов путем соединения. Если регистров нет, нужно указать тип `СправочникОбъект.ИдентификаторыОбъектовМетаданных` (для корректного заполнения свойств подписок на событие `ПроверитьДоступПередЗаписьюНабораЗаписей`, `ПроверитьДоступПриЗаписиНабораЗаписей`). Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`ОпределяемыйТип.ВладелецЗначенийКлючейДоступаНаборЗаписейРегистраРасчета`	Все регистры расчета с ограничением, а также регистры расчета, которые участвуют в ограничениях доступа других объектов путем соединения. Если регистров нет, нужно указать тип `СправочникОбъект.ИдентификаторыОбъектовМетаданных` (для корректного заполнения свойств подписок на событие `ПроверитьДоступПередЗаписьюНабораЗаписейРегистраРасчета`,

Свойство объекта метаданных	Настройка
	`ПроверитьДоступПриЗаписиНабораЗаписейРегистраРасчета`). Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`ОпределяемыйТип.ПолеРегистраКлючейДоступаКРегистрам`	Все типы собственных полей регистра, указанных в ограничении доступа к ним (типы опорных полей - аналог ссылки у объекта). Обновляется автоматически через `ПроверкаВнедренияБСП.erf`
`РегистрСведений.КлючиДоступаКРегистру<ИмяРегистраСОграничениемДоступа>`	Отдельный регистр сведений для хранения ключей доступа целевого регистра, рекомендуется создавать вместо использования общего поставляемого регистра `КлючиДоступаКРегистрам`, если в целевом регистре комбинаций значений полей, которые используются в ограничении доступа, может быть очень много (десятки тысяч и более). Для создания отдельного регистра нужно скопировать поставляемый регистр сведений `КлючиДоступаКРегистрам` и переименовать. После этого лишние измерения `Поле` * удалить, а состав типов оставшихся измерений `Поле` * указать одинаковый, полученный как сумма типов всех полей целевого регистра, которые используются в ограничении доступа

Далее перечислены определяемые типы, необходимые для стандартного варианта работы.

Свойство объекта метаданных	Настройка
`ОпределяемыйТип.ВладелецНаборовЗначенийДоступаОбъект`	• Типы объектов, для которых записываются на используемые как значения в проверках по в шаблон `#ПоЗначениямИНаборамРасширенный`. • Типы объектов с ограничением, настроенным `#ПоНаборамЗначений`

`ОпределяемыйТип.ВладелецВнешнихЗначенийВНаборахЗначенийДоступаОбъект`	Типы объектов, которые используются в процедур
	`ЗаполнитьНаборыЗначенийДоступа` для формирован
	в состав типов владельцев наборов значений дос
	типе `ОпределяемыйТип.ВладелецНаборовЗначенийДосту`
	`СправочникОбъект.Файлы`, используемый для форм
	`БизнесПроцессОбъект._ДемоЗаданиеСРолевойАдресаци`
	`УправлениеДоступомПереопределяемый.ПриИзмененииНа`

`ОпределяемыйТип.ВладелецСОграничениемПоНаборамЗначенийДоступаОбъект`	Используется для тех объектов (кроме документо наборы значений доступа в шаблоне `#ПоНаборамЗ` объекта

`ОпределяемыйТип.ВладелецСОграничениемПоНаборамЗначенийДоступаДокумент`	Используется для документов, которые использу
	доступа в шаблоне `#ПоНаборамЗначений` для огран
	документов нет, нужно указать тип
	`СправочникОбъект.ИдентификаторыОбъектовМетаданны`
	заполнения свойств подписки на событие
	`ЗаполнитьНаборыЗначенийДоступаТабличныхЧастейДоку`

`ТабличнаяЧасть.НаборыЗначенийДоступа.(НомерНабора, ЗначениеДоступа, Уточнение, Чтение, Изменение)`	Табличная часть создается в объектах метаданны конфигурации, которые используют наборы значе `#ПоНаборамЗначений` для ограничения доступа к ни Состав полей табличной части одинаков для всех типов – как у тех же полей в регистре сведений `Н`

Создание описаний поставляемых профилей групп доступа для начального заполнения и обновления информационной базы

Поставляемые профили групп доступа и папки задаются в процедуре `ЗаполнитьПоставляемыеПрофилиГруппДоступа` общего модуля `УправлениеДоступомПереопределяемый`. Они автоматически создаются при начальном заполнении, актуализируются при обновлении информационной базы, а также могут быть восстановлены администратором к первоначальному состоянию по этому описанию.

Для создания кода описания на встроенном языке рекомендуется применять обработку `УправлениеДоступом`, которая входит в дистрибутив библиотеки.

```
Процедура ЗаполнитьПоставляемыеПрофилиГруппДоступа(ОписанияПрофилей, ПараметрыОбновления) Экспорт

    // Папка профилей "Финансы".
    ЗаполнитьПапкуПрофилейФинансы(ОписанияПрофилей);

    // Профиль "Бухгалтер".
    ЗаполнитьПрофильБухгалтер(ОписанияПрофилей);
    // Профиль "Пользователь".
    ЗаполнитьПрофильПользователь(ОписанияПрофилей);
    ...

    // Дополнительные профили, которые не используются самостоятельно при настройке
    // прав пользователя, а дополняют основные профили, перечисленные выше.

    // Профиль "Ответственный за составы участников групп доступа (дополнительно)".
    ЗаполнитьПрофильОтветственныйЗаСоставыУчастниковГруппДоступа(ОписанияПрофилей);
    // Профиль "Ответственный за ведение номенклатуры (дополнительно)".
    ЗаполнитьПрофильОтветственныйЗаВедениеНоменклатуры(ОписанияПрофилей);
    ...

КонецПроцедуры
Процедура ЗаполнитьПапкуПрофилейФинансы(ОписанияПрофилей)

    ОписаниеПапки = УправлениеДоступом.НовоеОписаниеПапкиПрофилейГруппДоступа();
    ОписаниеПапки.Имя           = "Финансы";
    ОписаниеПапки.Идентификатор = "6849f943-d8c7-11eb-881f-b06ebfbf08c7";
    ОписаниеПапки.Наименование  =
        НСтр("ru = 'Финансы'", ОбщегоНазначения.КодОсновногоЯзыка());
    ОписанияПрофилей.Добавить(ОписаниеПапки);

КонецПроцедуры
Процедура ЗаполнитьПрофильБухгалтер(Знач ОписанияПрофилей)

    ОписаниеПрофиля = УправлениеДоступом.НовоеОписаниеПрофиляГруппДоступа();
    ОписаниеПрофиля.Имя           = "_ДемоБухгалтер";
    ОписаниеПрофиля.Родитель      = "Финансы";
    ОписаниеПрофиля.Идентификатор = "75fa0eca-98aa-11df-b54f-e0cb4ed5f655";
    ОписаниеПрофиля.Наименование  =
        НСтр("ru = 'Бухгалтер'", ОбщегоНазначения.КодОсновногоЯзыка());
    ОписаниеПрофиля.Описание =
        НСтр("ru = 'Предназначен для выполнения общих обязанностей бухгалтеров'");
    // Использование 1С:Предприятия.
    ОписаниеПрофиля.Роли.Добавить("ЗапускВебКлиента");
    ОписаниеПрофиля.Роли.Добавить("ЗапускТолстогоКлиента");
    ОписаниеПрофиля.Роли.Добавить("ЗапускТонкогоКлиента");
    ОписаниеПрофиля.Роли.Добавить("ВыводНаПринтерФайлБуферОбмена");
    ОписаниеПрофиля.Роли.Добавить("СохранениеДанныхПользователя");
    // Использование приложения.
    ОписаниеПрофиля.Роли.Добавить("БазовыеПраваБСП");
    ОписаниеПрофиля.Роли.Добавить("ПросмотрОписанияИзмененийПрограммы");
    ...
    // Основные возможности профиля.
    ОписаниеПрофиля.Роли.Добавить("_ДемоДобавлениеИзменениеБанковскихДокументов");
    ОписаниеПрофиля.Роли.Добавить("_ДемоДобавлениеИзменениеКассовыхДокументов");

    ОписанияПрофилей.Добавить(ОписаниеПрофиля);

КонецПроцедуры
```

> **Внимание!**
>
> Для того чтобы изменения вступили в силу сразу при разработке и отладке конфигурации, требуется обновить вспомогательные данные. См. раздел Обновление вспомогательных данных во время разработки.

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора приложения необходимо разместить:

- `Справочник.ГруппыДоступа` (кроме случая упрощенного интерфейса):
  - для частой работы администратора по просмотру и редактированию групп доступа (например, в подразделе **Важное** рядом со справочниками `Пользователи` и `ВнешниеПользователи`);
  - для частой работы администратора группы доступа по просмотру и редактированию участников групп доступа (например, в подразделе **Важное** рядом со справочником `Пользователи`).
- `Справочник.ПрофилиГруппДоступа` (кроме случая упрощенного интерфейса) – для редкой работы администратора по просмотру списка и содержания профилей групп доступа (например, в подразделе **См. также**).
- `ОграничиватьДоступНаУровнеЗаписей` – для включения/выключения ограничения доступа на уровне записей (администраторами). Если в ограничениях прав используются наборы значений доступа, то при включении константы необходимо выдавать предупреждение о необходимости заполнения данных для ограничения доступа.

- `Отчет.АнализПравДоступа` - для частой работы администратора по настройке прав пользователей (например, в подразделе **Важное** рядом со справочником `Пользователи`).

См. пример в форме `НастройкиПользователейИПрав` обработки `ПанельАдминистрированияБСП`.

Для размещения отчета `АнализПравДоступа` в формах списков объектов, в них необходимо выполнить встраивание подсистемы Подключаемые команды.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Управление доступом** следует использовать следующие роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Просмотр списка пользователей. Просмотр списка своих групп доступа в форме **Права доступа**. Формирование и просмотр отчета о своих правах доступа
2.	`ИзменениеУчастниковГруппДоступа` Изменение состава участников и просмотр тех групп доступа, у которых пользователь установлен как администратор группы доступа
3.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Просмотр и редактирование групп доступа (в том числе назначение администраторов групп доступа). Удаление объектов подсистемы, помеченных на удаление. Включение и отключение ограничений прав доступа на уровне записей. Разработка профилей групп доступа Использование обработки **Обновление доступа на уровне записей**. Ручной запуск регламентного задания **Заполнение данных для ограничения доступа**.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за составы участников групп доступа (дополнительно). Нужен для работы пользователя, которого назначили ответственным за состав участников в группе доступа (например, начальник подразделения, руководитель проектной группы и др.)	• `ИзменениеУчастниковГруппДоступа`, • `Подсистема_ДемоАдминистрирование` (просмотр подсистемы **Администрирование** подсистемы **Настройки программы**)

Особые случаи внедрения подсистемы

- Внедрение подсистем «Бизнес-процессы и задачи» и «Управление доступом».
- Внедрение подсистемы «Варианты отчетов» без подсистемы «Управление доступом».
- Внедрение подсистем «Взаимодействия» и «Управление доступом».
- Внедрение подсистемы «Дополнительные отчеты и обработки» без подсистемы «Управление доступом».
- Внедрение подсистемы «Заметки пользователя» без подсистемы «Управление доступом».
- Внедрение подсистем «Контроль ведения учета» и «Управление доступом».
- Внедрение подсистемы «Напоминания пользователя» без подсистемы «Управление доступом».
- Внедрение подсистемы «Пользователи» без подсистемы «Управление доступом».
- Внедрение подсистемы «Работа с почтовыми сообщениями» без подсистемы «Управление доступом».
- Внедрение подсистем «Работа с файлами» и «Управление доступом».
- Внедрение подсистемы «Рассылка отчетов» без подсистемы «Управление доступом».
- Внедрение подсистем «Свойства» и «Управление доступом».
- Внедрение подсистемы «Управление доступом» без необязательных подсистем.
- Внедрение подсистемы «Управление доступом» без подсистемы «Варианты отчетов».

Использование при разработке конфигурации

При создании новых прикладных объектов метаданных нужно повторно выполнять соответствующую часть процедуры настройки, описанной выше.

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Предопределенные виды доступа

В состав подсистемы входят виды доступа `Пользователи` и `ВнешниеПользователи`. Они позволяют сделать ограничение по типам значений доступа справочников `Пользователи`, `ГруппыПользователей` и `ВнешниеПользователи`, `ГруппыВнешнихПользователей` соответственно.

Могут применяться, например, для ограничения доступа к учетным записям электронной почты или к личным заказам клиентов.

При разработке логики ограничения следует иметь в виду, что в состав разрешенных пользователей вида доступа `Пользователи` автоматически включается текущий пользователь, аналогично в состав разрешенных внешних пользователей вида доступа `ВнешниеПользователи` автоматически включается текущий внешний пользователь.

Разработка оптимальных ограничений доступа

Эффективное наследование прав от объекта-владельца

Часто возникает задача организации доступа к подчиненным объектам, которые должны иметь те же права доступа, что и объект-владелец, т.е. они как бы «наследуют» права доступа владельца. Например, файлы-вложения, присоединенные к договору, можно просматривать и редактировать, только если сам договор доступен к просмотру и для редактирования. В таких случаях в тексте ограничения доступа применяется одна из функций `ЧтениеОбъектаРазрешено` или `ИзменениеОбъектаРазрешено`. Для реализации такой зависимости достаточно указать ограничение в виде:

```
Ограничение.Текст =
"РазрешитьЧтение
|ГДЕ
|    ЧтениеОбъектаРазрешено(ВладелецФайла)
|    ИзменениеОбъектаРазрешено(ВладелецФайла)";
```

Либо, например, для журналов документов:

```
Ограничение.Текст =
"РазрешитьЧтениеИзменение
|ГДЕ
|    ЧтениеОбъектаРазрешено(Ссылка)";
```

Подобные зависимости могут быть как одноуровневыми, так и многоуровневыми, когда, например документы накладных наследуют права заказов, а журнал документов наследует права накладных. Если зависимость одноуровневая и простая (без дополнительных условий), то обеспечивается максимальная производительность: встроенный оптимизатор использует поле `ВладелецФайла` в шаблоне `#ДляОбъекта(…)`, в результате чего не потребуется расчет права при записи зависимых объектов, а используются данные, уже подготовленные для владельца. При этом поле `ВладелецФайла` может быть также составного типа. С помощью инструмента разработчика `УправлениеДоступом.epf` можно увидеть используется ли поле владельца для оптимизации. Для всех остальных зависимостей возникают дополнительные затраты на пересчет прав доступа:

- На всех уровнях цепочки зависимостей, кроме последнего. Например, документ накладная наследует права заказа, а журнал документов наследует права накладной. В этом случае, для накладной произойдет пересчет прав, а для журнала это не потребуется.
- Если в ограничение добавлено любое дополнительное условие, например, по другим полям объекта.

Технически обновление (пересчет) прав на зависимые объекты выполняется после записи объекта-владельца в фоне в виде каскадной очереди:

- Во-первых, при обновлении прав владельца планируется обновление прав зависимых объектов (по возможности максимально «точечно», только на затронутые зависимые объекты, но при некоторых обстоятельствах обновление прав может стать массовым, например, если у владельца много зависимых объектов).
- Во-вторых, при перенастройке администратором прав в группах доступа планируется обновление прав зависимых объектов в разрезе таблиц (не точечно по отдельным объектам), что увеличивает нагрузку на фоновое обновление прав.
- В-третьих, возникает каскадная очередь обновления прав: сначала обновляются права владельцев, потом права зависимых объектов. Из-за этого права на зависимые объекты будут обновляться с некоторой задержкой, дольше обычного. Причем для старых объектов (по дате) эта задержка более значительна, чем для данных, введенных недавно.

Таким образом, без практической необходимости не рекомендуется создавать многоуровневые зависимости прав.

В некоторых случаях для сокращения количества уровней зависимостей можно указать поле владельца «через точку». Например, права на справочник `Файлы` наследует права от владельца по полю `ВладелецФайла`, а подчиненный справочник `ВерсииФайлов` - от справочника `Файлы` по полю `Владелец`. В данном случае для справочника `ВерсииФайлов` можно указать поле `Владелец.ВладелецФайлов`. Это несколько снизит производительность самого запроса RLS за счет дополнительного левого соединения, но позволит избежать каскадного пересчета прав на подчиненные элементы справочника `ВерсииФайлов`. Однако этот прием не рекомендуется, если реквизит `Владелец` составного типа или если права справочник `Файлы` уже зависят от другого вышестоящего объекта. В первом случае дополнительных левых соединений будет не одно, а уже несколько из-за составного типа, а во втором - такая оптимизация не приведет к снижению объема каскадного пересчета прав.

Эффективные зависимости при использовании других таблиц в ограничениях доступа

В ограничениях доступа к объекту можно присоединять произвольные таблицы, а также использовать обращения «через точку», например: `ЗначениеРазрешено(Владелец.Организация)`. Это удобный прием разработки, но в этом случае возникает зависимость от изменения значений полей других таблиц. Система будет автоматически отслеживать такие изменения и планировать фоновое обновление прав, что приведет к дополнительной нагрузке.

Степень этой нагрузки зависит от частоты изменения значений полей других таблиц, которые используются в логике ограничения доступа. Достаточно случаев, когда они обновляются очень редко, например, организация в договоре, скорее всего, никогда не поменяется и поэтому права на документы по договорам с организациями вообще не потребуется пересчитывать. В этом случае опасаться за производительность не следует. Несущественную нагрузку будут давать только сами запросы, проверяющие актуальность ключей доступа (проверка, что значения в ключах доступа соответствуют значениям, полученным из других таблиц).

Если же ожидается достаточно частое изменение данных в других таблицах, то рекомендуется пересмотреть структуру хранения данных или требования к ограничению доступа.

В любом случае не следует только ради более производительного обновления прав «механически» добавлять в зависимые объекты метаданных реквизиты-копии объектов-владельцев и реализовать их логику заполнения и обновления, т.к. механизм автоматического обновления прав доступа является многопоточным и, как правило, более эффективным, чем подобные частные реализации.

Эффективный объем служебных данных

Высокая скорость запросов RLS достигается за счет того, что права к объектам предварительно рассчитаны, хранятся как служебные данные в базе и извлекаются простыми запросами. Как правило, ориентировочно их объем составляет 2-10% от объема всей базы (в зависимости от сложности RLS, а также количества и состава данных).

Однако разработка сложных или неоптимальных ограничений доступа может привести к непропорциональному увеличению объема этих служебных данных. Это вызывает ряд негативных последствий:

- увеличивается время обновления прав (почти линейная зависимость от объема служебных данных);
- незначительно снижается производительность самих запросов с RLS;
- увеличиваются требования к месту на диске для самой базы и ее резервных копий.

По этой причине важно учитывать количество служебных данных, которое создает выбранное ограничение доступа, и стремиться по возможности применять оптимальные варианты.

1. Количество ключей доступа Ключи доступа – это элементы одноименного справочника `КлючиДоступа`, которые создаются автоматически для описания разных уникальных комбинаций значений доступа. Например, когда документы ограничиваются по организации и складам, тогда в ключ доступа помещаются комбинации организаций и складов, которые фактически встретились в этих документах. Ключи доступа сопоставляются с документами, у которых та же комбинация организации и склада по принципу «один ко многим». На ключи доступа рассчитываются права пользователей, которые сохраняются в служебных регистрах. Чем меньше ожидаемое количество ключей доступа на одну таблицу, тем более оптимально работают ограничения доступа. Условно малое количество ключей доступа составляет от 100 до 1000, среднее - от 1000 до 3000. Количество ключей можно оценить, исходя из следующего.
   - Количество уникальных значений в поле, которое проверяется в ограничении доступа. Например, если доступ к документу зависит от указанной в нем организации, то количество ключей доступа к документу не превышает количество всех организаций, введенных в базе.
   - Количество полей, которые участвуют в ограничении доступа. Например, если в ограничении доступа к документу проверяются организация и партнер, то в ключ доступа помещается комбинация организации и группы доступа партнеров. Чем больше различных сочетаний организаций с группами доступа партнеров в документе, тем больше ключей доступа.
   - Если используется табличная часть или другая таблица, тогда в соответствующую табличную часть ключа доступа помещаются все различные комбинации используемых полей. Например, если организация и партнер в табличной части документа, тогда в табличную часть ключа доступа попадут все различные комбинации организации и группы доступа партнеров. Чем больше различных наборов таких комбинаций в разных документах, тем больше ключей доступа.
   - Использование полей «через точку» (кроме полей табличных частей) равносильно использованию поля шапки, то есть значение помещается в шапку ключа. Если же присоединить таблицу по полю ссылка, тогда значение будет помещено в табличную часть ключа, что не увеличивает количество ключей, но увеличивает нагрузку на проверку ключей. (Поэтому лучше использовать описание «через точку», чем присоединять таблицу самостоятельно).
   - При проверке пользователя (или внешнего пользователя) в ключ доступа сохранится ссылка на пользователя. В данном случае невозможно использовать группы для уменьшения количества ключей доступа. Например, если в документе проверяется ответственный, тогда ключей доступа будет не меньше, чем различных ответственных, указанных во всех документах.
   Итого:
   - Использование меньшего числа проверок приводит к меньшему количеству ключей доступа в разы.
   - Использование групп значений доступа при настройке прав в группах доступа вместо самих значений (например, групп доступа партнеров, групп доступа номенклатуры вместо самих партнеров, номенклатуры), также приводит к существенно меньшему количеству ключей доступа, ориентировочно в 5-10 раз.
   - Использование реквизитов шапки приводит к меньшему количеству ключей и более простой и быстрой проверке актуальности ключей доступа.
   - В то же время, использование проверок доступа по пользователю (или внешнему пользователю), обычно существенно увеличивает количество ключей доступа, поэтому их следует применять только тогда, когда это действительно необходимо. Следует также иметь в виду, что количество ключей доступа зависит от настроек приложения. Отключение неиспользуемого ограничения по виду доступа в настройках приложения также приведет к сокращению общего количества ключей доступа (например, если отключить группы доступа номенклатуры). Это эффективно, когда проектируется универсальное прикладное решение с заранее большим количеством видов доступа, среди которых лишь небольшое подмножество будет фактически применяться на конкретном внедрении.
2. Количество записей прав доступа При расчете прав пользователей для каждого ключа доступа определяется наличие или отсутствие прав, т.е. устанавливается связь вида `КлючДоступа` - `Субъект`, где в качестве субъекта могут выступать группы доступа, наборы групп доступа, пользователи, внешние пользователи, наборы групп пользователей и наборы групп внешних пользователей. Технически, такая связь означает наличие, как минимум, права чтения и хранится в виде записи в служебном регистре. Количество связей зависит от объема данных в базе, от неоднородности данных (чем больше неоднородность, тем большее ключей доступа), от разнообразия настроек прав (от количества групп доступа и фактически используемых наборов групп доступа), а также от количества пользователей. Например, если ключей доступа 100 тыс., а пользователей 5 тыс., тогда в пределе (когда у пользователей есть почти все права) получится 500 миллионов связей, что сможет обработать только очень мощный сервер. Ориентировочно, для среднего сервера оптимальное количество таких связей не более 5 млн., для мощного сервера - до 50 млн. Для небольших прикладных решений со слабым сервером следует ориентироваться до 0.5 млн. связей. При проектировании ограничений доступа не рекомендуется использовать ограничения по пользователям (внешним пользователям) без реальной необходимости, так как расчет прав на такие ключи доступа всегда идет с точностью до пользователя, что значительно увеличивает количество записей прав доступа.

3. Количество групп записей регистров Связь ключей доступа с данными хранится в служебных регистрах сведений `КлючиДоступаКОбъектам` и `КлючиДоступаКРегистрам`. В нормальном режиме работы в регистре `КлючиДоступаКОбъектам` хранится столько записей, сколько ссылок в базе данных. Также в нормальном режиме в служебном регистре `КлючиДоступаКРегистрам` должно быть количество записей соизмеримое с количеством записей в регистре `КлючиДоступаКОбъектам`, для того чтобы достаточно быстро выполнялось обновление прав, а также сами запросы с RLS. В отличие от ключей доступа, рассчитанных для ссылочных данных, ключи доступа к регистрам формируются и связываются с наборами данных иначе. Каждый раз при записи набора записей регистра в регистр сведений `КлючиДоступаКРегистрам` добавляются новые комбинации значений опорных полей, которые прямо или косвенно участвуют в ограничении доступа к регистру. Чем больше разнообразие таких комбинаций, тем больше строк в регистре `КлючиДоступаКРегистрам`. Например, если доступ ограничен по измерениям регистра `Организация`, `Подразделение` и «через точку» `АналитикаУчетаНоменклатуры.Партнера`, `АналитикаУчетаНоменклатуры.МестоХранения`, `АналитикаУчетаНоменклатуры.Номенклатура`, то опорных полей, комбинации значений которых условно делят записи регистра на группы с точки зрения одинакового доступа, всего три: `Организация`, `Подразделение`, `АналитикаУчетаНоменклатуры`. Например, если опорное поле всего одно и разных значений в нем мало, например, поле со ссылкой на справочник `Организации`, то беспокоится не о чем. Однако если же в ограничении доступа к регистру используется поля-ссылки на документы, то в регистр `КлючиДоступаКРегистрам` добавится очень много строк (вероятно, сколько в исходном регистре), что существенно замедлит запросы с RLS не только к этому регистру, но и к любым другим регистрам с RLS. В этом случае следует найти другое проектное решение, либо создать отдельный служебный регистр сведений `КлючиДоступаКРегистру<ИмяРегистра>` вместо общего регистра `КлючиДоступаКРегистрам`. В таком случае большой объем служебных данных к одному регистру с неоптимальным RLS не будет снижать производительность всех остальных регистров. Ограничение доступа к регистрам, использующим общий регистр сведений `КлючиДоступаКРегистрам`, можно считать производительным, если в нем в сумме будет до 10 млн. строк из расчета на средний сервер.

Нагрузочное тестирование

Выбранный вариант решения рекомендуется проверять на нагрузочных тестах прикладного решения, запуская тест, когда ограничение доступа на уровне доступа включено и когда выключено, чтобы оценить приемлемость дополнительных задержек от RLS.

Нагрузочные тесты рекомендуется делать по технологии моделирования многопользовательской работы сотрудников, выделяя ключевые операции и определяя количество операций в минуту на одного пользователя, а также допустимые задержки на ключевые операции. В результате тестирования получится APDEX по ключевым операциям в соответствии с планом допустимых задержек, который легко использовать для оценки производительности прикладного решения.

- Количество потоков обновления доступа иногда может существенно влиять на производительность (когда много ограничений с зависимыми объектами). При интенсивности добавления/изменения ведущих объектов, требующих обновления доступа зависимых объектов (то есть, когда добавляется точечное задание обновления доступа):
  - до 10 в минуту достаточно 2-х потоков,
  - до 50 в минуту - 5 потоков,
  - до 100 в минуту - 10 потоков,
  - до 250 в минуту - 20 потоков.
- Если тактовая частота процессора низкая 2-2.5 ГГц, то число потоков может потребоваться больше в 1.5-2 раза. При высокой интенсивности изменений лучше использовать процессор с большой тактовой частотой 4-5 ГГц, а не увеличивать количество потоков, так как после 40 потоков дальнейшее увеличение производительности за счет числа потоков может быть недостаточным из-за общих задержек.

Для организации регулярного нагрузочного тестирования хорошо подходит Тест-центр конфигурации 1С:Корпоративный инструментальный пакет.

Регулярный запуск отчета ПроверкаВнедренияБСП

С помощью отчета `ПроверкаВнедренияБСП.erf` рекомендуется регулярно выполнять автоматическую проверку и обновление

- используемых вариантов шаблонов с их параметрами в ролях,
- внутренней реализации стандартных шаблонов и их наличия в соответствии с фактическим использованием в ролях,
- состава определяемых типов и вспомогательных предопределенных элементов справочников `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений`.

Для этого в отчете установить отбор для проверки по подсистеме **Управление доступом** и режим исправления ошибок с отметкой всех ошибок подсистемы **Управление доступом**. При исправлении ошибок в ролях существующие ограничения в формате БСП 2.0, например:

```
#ПоЗначениям(… "Организации", "Организация", …)
```

не изменяются, а «оборачиваются» обязательным стандартным условием:

```
#Если &ОграничениеДоступаНаУровнеЗаписейУниверсально #Тогда
#ДляОбъекта(…)
#Иначе
#ПоЗначениям(… "Организации", "Организация", …)
#КонецЕсли
```

Для сведения:

- Вспомогательные предопределенные элементы справочников `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений` используются в качестве первого параметра шаблона `ДляРегистра`, к ним выполняется обращение по имени: `<Имя справочника>.<Полное имя регистра без точки>`.
- Допустимо, что вспомогательные предопределенные элементы в справочниках `ИдентификаторыОбъектовМетаданных` и `ИдентификаторыОбъектовРасширений` могут быть удалены и созданы заново, то есть иметь разные внутренние идентификаторы платформы

1С:Предприятие (без потери данных и нарушения работоспособности). Избыточные (мусорные) предопределенные элементы также не влияют на корректную работу конфигурации.

- Не следует использовать вспомогательные предопределенные элементы в коде. Вместо этого следует использовать функцию `ИдентификаторОбъектаМетаданных` общего модуля `ОбщегоНазначения`. В дальнейшем поддержка этих вспомогательных предопределенных элементов может быть прекращена.

Проверка совпадения шаблонов и ограничений доступа в разных ролях

Шаблон ограничения доступа должен совпадать во всех ролях. Ограничение доступа для одного и того же права одной и той же таблицы также должно совпадать во всех ролях, если указано.

Для сравнения вручную нужно в конфигураторе:

- захватить роли, в которых будет осуществляться проверка, если конфигурация подключена к хранилищу;
- хотя бы временно установить ролям правило поддержки **Редактируется с сохранением поддержки**, если роли находятся на поддержке у конфигурации поставщика;
- открыть окно **Все ограничения доступа**;
- если сравниваются ограничения доступа, тогда остаться на вкладке **Ограничения доступа**, если сравниваются шаблоны, тогда перейти на вкладку **Шаблоны ограничений**;
- выделить две или более строк, в которых требуется сравнить тексты и нажать кнопку **Изменить текущий элемент (F2)**.
- при сравнении ограничений откроется окно **Ограничение доступа**, а при сравнении шаблонов откроется окно **Изменение шаблона ограничений**;
- если текст в открывшемся окне пустой, тогда он не совпадает, если не пустой, значит совпадает.

> **Примечание**
>
> Кнопка **Изменить текущий элемент (F2)** будет заблокирована, если хотя бы одна роль недоступна для редактирования среди выделенных, кроме того, кнопка может быть доступна, но не срабатывать при нажатии, если у конфигурации режим совместимости более 8.3.12 (обход - переключить режим совместимости на 8.3.12 на время обновления шаблонов, а потом вернуть обратно).

Для автоматизированного сравнения нужно:

- закрыть конфигуратор (для файловой ИБ закрывать не нужно);
- запустить **1С:Предприятие**, открыть отчет `ПроверкаВнедренияБСП.epf`;
- установить отбор **Проверять внедрение подсистем**: отметить только **Управление доступом**;
- нажать **Сформировать** - отчет покажет все ошибки подсистемы **Управление доступом**, в том числе различия текстов шаблонов.

Копирование шаблонов ограничения доступа при разработке

Чтобы добавить новое ограничение с использованием шаблона требуется скопировать шаблон из роли `ИзменениеУчастниковГруппДоступа` в свою роль. Для этого нужно:

- открыть роль `ИзменениеУчастниковГруппДоступа` и перейти на вкладку **Шаблоны ограничений**, выделить шаблон, который требуется скопировать и скопировать его наименование из ячейки в буфер обмена;
- открыть свою роль, перейти на вкладку **Шаблоны ограничений**, добавить новую строку и ввести наименование из буфера обмена;
- переключиться в окно роли `ИзменениеУчастниковГруппДоступа`, нажать кнопку **Установить текст шаблона**, в окне **Ограничение доступа** выделить весь текст шаблона и скопировать в буфер обмена;
- переключиться в окно своей роли, нажать кнопку **Установить текст шаблона**, в окне **Ограничение доступа** вставить текст из буфера обмена;

> **Примечание**
>
> Копирование текста шаблона в буфер обмена следует производить только из отдельного окна установки/изменения текста шаблона, а не из ячейки списка шаблонов, в противном случае непечатаемые символы могут быть изменены и тексты не будут точно совпадать. Аналогично и для текста ограничения.

Обновление шаблонов ограничения доступа в ролях

При изменении шаблонов ограничения доступа, поставляемых в роли `ИзменениеУчастниковГруппДоступа` (`ДляОбъекта`, `ДляРегистра`, `ПоЗначениям`, `ПоНаборамЗначений`, `ПоЗначениямРасширенный`, `ПоЗначениямИНаборамРасширенный`) требуется обновить их во всех ролях конфигурации без исключения (то есть, включая библиотеки).

Если конфигурация подключена к хранилищу, тогда все роли нужно захватить (в конфигураторе в окне **Хранилище конфигурации** можно выделить роли массово `Shift` + `PageDown` и один раз захватить).

Если конфигурация находится на поддержке библиотек, тогда все роли всех библиотек нужно перевести в режим **Редактируется с сохранением поддержки** (в конфигураторе в окне **Настройка поддержки** можно развернуть вершину **Роли**, выделить роли массово `Shift` + `PageDown` и один раз установить правило поддержки). При обновлении библиотеки устанавливать правило для новых ролей библиотек, делать замещение всех измененных ролей на роли, поставляемые библиотекой, а затем повторно обновлять шаблоны.

Для обновления вручную нужно в конфигураторе:

- открыть роль `ИзменениеУчастниковГруппДоступа` и перейти на вкладку **Шаблоны ограничений**, выделить первый шаблон, который требуется обновить, нажать кнопку **Установить текст шаблона**, в окне **Ограничение доступа** выделить весь текст шаблона и скопировать в буфер обмена;

- открыть окно **Все ограничения доступа** и перейти на вкладку **Шаблоны ограничений**, нажать кнопку **Отбор**, снять флажок с роли `ИзменениеУчастниковГруппДоступа`, в поле **Наименование** выбрать первый шаблон, который требуется обновить и нажать **ОК**; выделить все роли (CTL+A), нажать кнопку **Изменить текущий элемент (F2)** - откроется окно **Изменение шаблона ограничений**, снять флажок **Изменять наименование**, установить флажок **Изменять текст**, вставить текст шаблона из буфера обмена и нажать **ОК**.
- выполнить предыдущие два пункта для всех шаблонов, которые требуется обновить.

> **Примечание**
>
> - если в окне **Изменение шаблона ограничений** текст не пустой, тогда он совпадает у выделенных ролей.
> - кнопка **Изменить текущий элемент (F2)** будет заблокирована, если хотя бы одна роль недоступна для редактирования среди выделенных, кроме того, кнопка может быть доступна, но не срабатывать при нажатии, если у конфигурации режим совместимости более 8.3.12 (обход - переключить режим совместимости на 8.3.12 на время обновления шаблонов, а потом вернуть обратно).
> - копирование текста шаблона в буфер обмена следует производить только из отдельного окна установки/изменения текста шаблона, а не из ячейки списка шаблонов, в противном случае непечатаемые символы могут быть изменены и тексты не будут точно совпадать. Это приведет к отображению пустого текста при открытии окна **Изменение шаблона ограничений** для выделенных строк шаблонов и обнаружению расхождения шаблонов отчетом `ПроверкаВнедренияБСП.erf`. Аналогично и для текста ограничения.

Для автоматического обновления нужно:

- закрыть конфигуратор, запустить **1С:Предприятие**, открыть отчет `ПроверкаВнедренияБСП.epf`;
- установить отбор **Проверять внедрение подсистем**: отметить только **Управление доступом**;
- установить отбор **Исправлять ошибки**: отметить только две ошибки **Управление доступом. Используемый шаблон в роли отсутствует или отличается от поставляемого**, **Управление доступом. Шаблон не используется в роли**;
- сформировать отчет (и сохранить, для возможности в дальнейшем уточнить, что было исправлено);
- закрыть **1С:Предприятие**, открыть конфигуратор сравнить изменения основной конфигурации с конфигурацией базы данных, убедится, что нет лишних изменений и зафиксировать результат (поместить изменения в хранилище, обновить конфигурацию базы данных).

> **Примечание**
>
> Отчет `ПроверкаВнедренияБСП.epf` имеет функцию `ПроверитьВнедрение` в модуле объекта в области программного интерфейса, которая позволяет автоматизировать процесс обновления шаблонов, а также исправления других ошибок подсистемы **Управление доступом**.

Обеспечение производительности пакетной обработки данных (полноправный пользователь)

Когда константа `ОграничиватьДоступНаУровнеЗаписейУниверсально` включена, тогда при изменении объектов (или наборов записей), в том числе в режиме `ОбменДанными.Загрузка = Истина`, происходит обновление ключей доступа к объектам (или к регистрам), которое необходимо для корректной проверки прав доступа для неполноправных пользователей. Это добавляет, как правило, 5–20 мс на 1 объект или набор записей, что при записи большого числа «легких» объектов может быть существенным замедлением. Чтобы сократить эти затраты, следует отключить обновление ключей доступа на период загрузки, а затем включить (при включении произойдет планирование обновления ключей доступа для таблиц, в которые внесены изменения с момента отключения). Таким образом общие затраты на загрузку будут снижены, а обновление ключей доступа будет выполнено в фоне, что более эффективно, если объектов много, так как в фоне обновление ключей доступа выполняется пакетно, а не по одному. Для отключения обновления ключей доступа предусмотрена процедура `ОтключитьОбновлениеКлючейДоступа` общего модуля `УправлениеДоступом` (примеры см. в комментарии к процедуре).

Проверка работы логики ограничений прав

Ошибка доступа к данным связана либо с недостаточным, либо избыточным доступом к данным информационной базы. В общем виде подобные нештатные ситуации могут возникать по причинам, перечисленным ниже.

- Возможная ошибка при настройке прав доступа:
  - некорректная настройка состава пользователей или групп пользователей группы доступа – необходимо включить пользователя или группу пользователей в группы доступа или исключить его из них;
  - некорректный профиль группы доступа – выбрать другой профиль групп доступа, создать дополнительный профиль или скорректировать имеющийся профиль;
  - некорректная настройка ограничений доступа в группе доступа (списков разрешенных значений, например: организаций, групп партнеров, складов) – необходимо добавить или исключить значение доступа из группы доступа.
- Возможная ошибка при разработке или доработке конфигурации на внедрении (в компетенции разработчика конфигурации):
  - некорректная настройка состава ролей в профиле;
  - некорректная настройка состава видов доступа в профиле.
- Ошибки разработчика конфигурации:
  - ошибка в параметрах стандартного шаблона;
  - ошибка в составе типов подписок на события;
  - ошибка в описании видов доступа;
  - ошибка в логике ограничения доступа в процедуре `ПриЗаполненииОграниченияДоступа`;
  - ошибка в процедуре заполнения наборов значений доступа (стандартный вариант).
- Ошибки разработчика этой подсистемы:
  - ошибка в стандартном шаблоне;
  - ошибка из-за особенностей работы СУБД;
  - ошибка в общих модулях подсистемы.

При диагностике ошибок доступа к данным также нужно проверить следующее:

- Если в профиль не добавлена роль для чтения каких-либо данных, но предусмотрены роли для работы с другими связанными объектами, то возможна ситуация, когда в полях этих объектов будут отображаться значения вида **Ссылка не найдена (… : …)** вместо самих данных.
- В стандартном варианте убедиться, что содержимое регистра сведений **Наборы значений доступа** содержит наборы значений доступа, которые записаны таким образом, что наборы с одинаковыми номерами для одного объекта соответствуют логике ограничения доступа по условию `И` а затем наборы с разными номерами «складываются» по условию `ИЛИ` :
  - если по интересующему нас объекту в регистре сведений нет записей, то объект не будет доступен для всех пользователей;
  - если по интересующему нас объекту нет ни одного набора, относящегося к конкретному праву доступа, значит, для всех пользователей нет этого права на объект.

Разработка обработчиков обновления ИБ

В процессе разработки конфигурации в нее могут быть внесены изменения, которые потребуют обновления данных для ограничения доступа. Ниже перечислены типовые ситуации, когда обновление необходимо.

1. При каждом обновлении информационной базы выполняются обработчики обновления в модуле `УправлениеДоступомСлужебный` :
   - Для обновления неразделенных вспомогательных данных (кешей свойств встраивания), которые сохраняются в регистре сведений `ПараметрыРаботыПрограммы` , вызывается процедура `ОбновитьПараметрыОграниченияДоступа` . Процедура обновляет общие параметры, права ролей, зависимости прав доступа, описания поставляемых профилей, состав предопределенных профилей, описание возможных прав для настройки прав объектов, например:
     - при добавлении объектов метаданных (доступ к которым может ограничиваться) и добавлении/изменении ролей обновляется регистр сведений `ПраваРолей` и регистрируется состав объектов метаданных, для которых внесены изменения;
     - при изменении описаний поставляемых профилей групп доступа регистрируются их изменения;
     - при изменении зависимостей прав доступа между объектами метаданных регистрируются их изменения.
   - Для обновления разделенных вспомогательных данных (разных для каждой области данных) вызывается процедура `ОбновитьВспомогательныеДанныеПоИзменениямКонфигурации` . В том числе обновляется регистр сведений `ПараметрыОграниченияДоступа` . Соответственно, в процессе разработки необходимо учитывать, что для «вступления в силу» перечисленных изменений необходим запуск обработчиков (перед проверкой текущей разработки). При этом если был изменен только текст ограничения в модуле менеджера, то при попытке изменить этот объект подсистема автоматически обновит параметры ограничения доступа. Обычно для обновления требуется изменить версию конфигурации на следующую. Для повторных запусков обработчиков следует изменять версию в регистре сведений `ВерсииПодсистем` на предыдущую.

> **Внимание!**
>
> В процессе разработки рекомендуется использовать обработку `ОбновлениеВспомогательныхДанных.epf` , которая входит в состав дистрибутива библиотеки. При закладке в хранилище результатов разработки, требующих обновления вспомогательных данных, рекомендуется поместить в хранилище новую версию конфигурации, чтобы обновление информационной базы выполнилось у всех участков разработки.

2. **Обновление поставляемых профилей групп доступа**. Если в конфигурации было изменено описание поставляемых профилей групп доступа в процедуре `ПриЗаполненииПоставляемыхПрофилейГруппДоступа` общего модуля `УправлениеДоступомПереопределяемый` , то никаких действий не требуется. При обновлении информационной базы обновление поставляемых профилей будет выполнено автоматически. Для управления обновлением предусмотрены свойства в параметре процедуры `ПараметрыОбновления` . Их значения можно переопределить (подробнее см. комментарий к процедуре).
3. **Начальное заполнение** при включении ограничений на уровне записей требуется:
   - если среди видов доступа применяются группы значений доступа, тогда требуется выполнить заполнение групп значений доступа;
   - в стандартном варианте, если в ролях конфигурации имеются ограничения прав, которые используют наборы значений доступа. Для выполнения начального заполнения следует воспользоваться одним из двух способов:
   - Разрешить выполнение регламентного задания `ЗаполнениеДанныхДляОграниченияДоступа` ; этот способ позволяет выполнить начальное заполнение наборов значений доступа порциями в фоне, не мешая работе администратора. Однако для того, чтобы изменения вступили в силу и с базой смогли начать работу другие пользователи, необходимо дождаться завершения отработки регламентным заданием всех порций.
   - Вызвать процедуру `ЗаполнениеДанныхДляОграниченияДоступа` общего модуля `УправлениеДоступом` с параметром `КоличествоДанных` = 0.

> **Примечание:**
>
> Операция обновления наборов значений доступа соизмерима с перезаписью всех документов, доступ к которым ограничивается на уровне записей, поэтому процесс обновления на большой базе может занять достаточно много времени. В связи с этим лучше использовать предусмотренное регламентное задание. Для этого достаточно включить его использование (отключение произойдет автоматически).

4. **Обновление доступа** (производительный вариант) выполняется автоматически. Ход обновления можно проконтролировать в форме **Обновление доступа на уровне записей** (раздел **Администрирование** – **Настройки пользователей и прав**).
5. В стандартном варианте **обновление наборов значений доступа** необходимо выполнять в случаях:
   - Если в конфигурации были добавлены виды доступа, использующие группы значений доступа, или изменены существующие виды доступа так, что они стали использовать группы значений доступа, то необходимо выполнить обновление наборов значений доступа. Для этого следует воспользоваться методикой, описанной в пункте 3.
   - Если в конфигурации у определенных объектов метаданных был изменен алгоритм формирования наборов значений доступа или были изменены зависимости по правам, то для решения этой задачи следует выбрать один из двух вариантов:
     - перезаписать наборы значений доступа с помощью процедуры `ОбновитьНаборыЗначенийДоступа` общего модуля `УправлениеДоступом` ;
     - очистить наборы значений доступа для соответствующих объектов и заполнить их способом, который описан в пункте 3.

Настройка форм выбора значений доступа

Иногда, формы выбора элементов работают нестандартно. При выборе значений доступа в одних формах требуется установить дополнительный отбор, а в других наоборот, отключить лишний отбор. Для реализации этой логики в параметры формы выбора пробрасывается специальный параметр `ЭтоВыборЗначенияДоступа`, наличие которого можно использовать как признак для реализации дополнительной логики.

Особенности работы отчета "Анализ прав доступа"

Отчет **Анализ прав доступа** помимо информации об уровне доступа к таблицам и отчетам может получать информацию о таблицах, используемых в том или ином отчете. При этом в некоторых отчетах необходимо явно указывать таблицы, которые в них используются. В таком случае информация о таблицах в отчете **Анализ прав доступа** будет выводиться корректно.Подробнее см. в разделе Варианты отчетов.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме:

- константа `КоличествоПотоковОбновленияДоступа`,
- константа `ПоследнееОбновлениеДоступа`,
- регистр сведений `ОбновлениеКлючейДоступаТекущиеЗадания`.

Из планов обмена распределенной информационной базы (РИБ) и автономного рабочего места, в которых предусмотрен фильтр по организациям, подразделениям и другим объектам, рекомендуется также исключать следующие объекты метаданных:

- справочник `КлючиДоступа`,
- справочник `НаборыГруппДоступа`,
- регистр сведений `КлючиДоступаКОбъектам`,
- регистр сведений `КлючиДоступаКРегистрам`,
- регистры сведений `КлючиДоступаКРегистру`,
- регистр сведений `КлючиДоступаГруппДоступа`,
- регистр сведений `КлючиДоступаНаборовГруппДоступа`,
- регистр сведений `КлючиДоступаПользователей`,
- регистр сведений `КлючиДоступаВнешнихПользователей`,
- регистр сведений `ОбновлениеКлючейДоступаКДанным`,
- регистр сведений `ОбновлениеКлючейДоступаПользователей`.

В планах обмена распределенной информационной базы (РИБ) рекомендуется отключать регистрацию изменений для следующих объектов метаданных подсистемы (см. также раздел Особенности создания начального образа подчиненного узла распределенной ИБ):

- константа `ПервоеОбновлениеДоступаЗавершилось`,
- справочник `КлючиДоступа`,
- справочник `НаборыГруппДоступа`,
- регистр сведений `КлючиДоступаКОбъектам`,
- регистр сведений `КлючиДоступаКРегистрам`,
- регистры сведений `КлючиДоступаКРегистру`,
- регистр сведений `КлючиДоступаГруппДоступа`,
- регистр сведений `КлючиДоступаНаборовГруппДоступа`,
- регистр сведений `КлючиДоступаПользователей`,
- регистр сведений `КлючиДоступаВнешнихПользователей`,
- регистр сведений `ОбновлениеКлючейДоступаКДанным`,
- регистр сведений `ОбновлениеКлючейДоступаПользователей`,
- регистр сведений `ПараметрыОграниченияДоступа`,
- регистр сведений `ПраваРолей`,
- регистр сведений `ЗависимостиПравДоступа`,
- регистр сведений `ИспользуемыеВидыДоступаПоТаблицам`,
- регистр сведений `ТаблицыГруппДоступа`,
- регистр сведений `ГруппыЗначенийДоступа`,
- регистр сведений `ЗначенияГруппДоступа`,
- регистр сведений `ЗначенияГруппДоступаПоУмолчанию`.

Из планов обмена, предназначенных для синхронизации данных между различными приложениями, рекомендуется исключать следующие объекты метаданных:

- константа `КоличествоПотоковОбновленияДоступа`,
- константа `ПоследнееОбновлениеДоступа`,
- справочник `КлючиДоступа`,
- справочник `НаборыГруппДоступа`,
- регистр сведений `КлючиДоступаКОбъектам`,
- регистр сведений `КлючиДоступаКРегистрам`,
- регистры сведений `КлючиДоступаКРегистру`,
- регистр сведений `КлючиДоступаГруппДоступа`,
- регистр сведений `КлючиДоступаНаборовГруппДоступа`,
- регистр сведений `КлючиДоступаПользователей`,
- регистр сведений `КлючиДоступаВнешнихПользователей`,
- регистр сведений `ОбновлениеКлючейДоступаКДанным`,

- регистр сведений `ОбновлениеКлючейДоступаПользователей`,
- регистр сведений `ОбновлениеКлючейДоступаТекущиеЗадания`,
- регистр сведений `ПараметрыОграниченияДоступа`,
- регистр сведений `ПраваРолей`,
- регистр сведений `ЗависимостиПравДоступа` (стандартный вариант),
- регистр сведений `ИспользуемыеВидыДоступаПоТаблицам`,
- регистр сведений `ТаблицыГруппДоступа`,
- регистр сведений `ГруппыЗначенийДоступа`,
- регистр сведений `ЗначенияГруппДоступа`,
- регистр сведений `ЗначенияГруппДоступаПоУмолчанию`.

Управление итогами и агрегатами

Подсистема **Управление итогами и агрегатами** предоставляет средства для администрирования итогов и агрегатов регистров информационной базы. Подсистема позволяет как выполнять типовые административные операции, так и дает доступ к следующим возможностям:

- включение/отключение использования итогов и агрегатов,
- разделение итогов,
- установка периода и пересчет итогов,
- перестроение и обновление агрегатов,
- расчет оптимальных агрегатов.

Настройка

После переноса объектов подсистемы в конфигурацию необходимо установить расписание регламентных заданий, а также флажок **Использование**:

- `ОбновлениеАгрегатов`,
- `ПерестроениеАгрегатов`,
- `УстановкаПериодаРасчитанныхИтогов`.

Размещение в командном интерфейсе

Для использования подсистемы необходимо разместить в командном интерфейсе администратора объект метаданных – обработку `УправлениеИтогамиИАгрегатами` для использования всех возможностей подсистемы.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Управление итогами и агрегатами** следует использовать роли:

№	Роли и их назначение
1.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Доступ к администрированию итогов и агрегатов (доступ к обработке).

Использование при разработке конфигурации

При разработке конфигурации никаких дополнительных действий не требуется.

Настройка обмена данными

Для настройки обмена данными никаких действий не требуется, так как подсистема не предоставляет собственных данных.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между различными программами. Подсистема рассчитана на работу только в одном узле, поэтому в различных узлах данные подсистемы должны храниться независимо.

Учет оригиналов первичных документов

Подсистема **Учет оригиналов первичных документов** предназначена для отслеживания состояний напечатанных первичных документов, например, в сценарии: документ распечатали, передали клиенту, ждем возвращения подписанного оригинала. Подсистему можно подключить к произвольным документам конфигурации, входящим, исходящим и двусторонним первичным документам. При этом каждый из документов в информационной системе может иметь несколько печатных форм.

Для использования подсистемы в прикладной конфигурации также должен иметься или быть разработан общий журнал документов, т.е. реестр документов. Пример см. `_ДемоРеестрДокументов`.

Настройка

Размещение в командном интерфейсе

Если в конфигурации не используется подсистема **Настройки программы**, то в командном интерфейсе администратора необходимо разместить

- константу `ИспользоватьУчетОригиналовПервичныхДокументов` – для включения и выключения подсистемы,
- и поместить справочник `СостоянияОригиналовПервичныхДокументов` – для настройки используемых состояний оригиналов.

См. пример в форме `ПечатныеФормыОтчетыИОбработки` обработки `ПанельАдминистрированияБСП`.

Если планируется отдельное рабочее место, то необходимо определить требуемые разделы командного интерфейса, в которые разместить общую команду `ЖурналУчетаОригиналовПервичныхДокументов`. Рекомендуется выбрать все разделы, в которых предусмотрена работа с документами.

Определение состава документов

Необходимо определить состав документов, для которых требуется вести учет оригиналов. Например, документы по реализации и приобретению услуг и товаров, корректировки реализации и приобретений, документы переработки. Эти документы также должны входить в состав реестра документов. Пример состава документов см. в типе измерения `Ссылка` регистра сведений `_ДемоРеестрДокументов`.

Далее документы, являющиеся поставщиками оригиналов первичных документов, следует указать в составе определяемого типа `ОбъектСУчетомОригиналовПервичныхДокументов`.

Формы списков, в которые должны выводиться команды учета оригиналов первичных документов, следует перечислить в процедуре `ПриОпределенииОбъектовСКомандамиУчетаОригиналов` общего модуля `УчетОригиналовПервичныхДокументовПереопределяемый`, пример:

```
Процедура ПриОпределенииОбъектовСКомандамиУчетаОригиналов(СписокОбъектов) Экспорт
    СписокОбъектов.Добавить("Документ._ДемоРеализацияТоваров.Форма.ФормаСписка");
    СписокОбъектов.Добавить("Документ._ДемоПеремещениеТоваров.Форма.ФормаСписка");
    СписокОбъектов.Добавить("Обработка._ДемоЖурналУчетаОригиналовПервичныхДокументов.Форма.СписокДокументов");
КонецПроцедуры
```

Подсистема может отслеживать состояния первичных документов по определенным печатным формам. Необходимо определить состав печатных форм документов, для которых требуется вести учет оригиналов. Например, у документа "Отпуск" может быть 4 печатных формы, но только в одной требуется подпись сотрудника и, соответственно, только по ней требуется отслеживать состояние.

Далее документы, являющиеся поставщиками оригиналов первичных документов, и печатные формы по которым требуется отслеживание необходимо перечислить в процедуре `ЗаполнитьТаблицуУчетаОригиналов` модуля `УчетОригиналовПервичныхДокументовПереопределяемый`, пример:

```
Процедура ЗаполнитьТаблицуУчетаОригиналов(ТаблицаУчетаОригиналов) Экспорт
    НоваяСтрока = ТаблицаУчетаОригиналов.Добавить();
    НоваяСтрока.ОбъектМетаданных = Метаданные.Документы._ДемоРеализацияТоваров;
    НоваяСтрока.Идентификатор = "РасходнаяНакладная";
КонецПроцедуры
```

При этом, если документ подключен к подсистеме учета оригиналов и в данном переопределяемой процедуре:

- ничего не указывать по этому документу, то состояния будут отслеживаться по всем печатным формам этого документа;
- указать печатные формы, то состояния будут отслеживаться только по указанным печатным формам документа.

Поддерживается отслеживание состояний по "многосотрудниковым" документам. Пример см. документ `_ДемоОтпускаСотрудников`, в печатной форме `"Приказ о предоставлении отпуска работникам (Т-6а)"` требуется подпись нескольких сотрудников (они хранятся в табличной части документа), поэтому состояния для таких документов отслеживается и в разрезе сотрудников.

Для подключения таких документов к учету оригиналов необходимо в составе определяемого типа `Сотрудник` указать соответствующие справочники. Как правило, это такие справочники, как **Физические лица**, **Сотрудники**. Необходимо определить состав документов, для которых требуется подпись нескольких сотрудников, и перечислить их и наименование табличной части, в которой хранится список сотрудников, в процедуре `ПриОпределенииМногосотрудниковыхДокументов` общего модуля `УчетОригиналовПервичныхДокументовПереопределяемый`, например:

```
Процедура ПриОпределенииМногосотрудниковыхДокументов(СписокОбъектов) Экспорт
    СписокОбъектов.Вставить(Метаданные.Документы._ДемоОтпускаСотрудников.ПолноеИмя(),"Сотрудники");
КонецПроцедуры
```

При этом, если документ подключен к подсистеме учета оригиналов и в данной переопределяемой процедуре:

- не указывать такой документ, то состояния будут отслеживаться в разрезе печатных форм, как и раньше;
- указать документ и табличную часть, то состояния будут отслеживаться по печатным формам документа в разрезе сотрудников.

Подключение форм документов

Для подключения форм документов, в которых требуется выводить сведения о текущем состоянии оригинала первичного документа необходимо в формах документов, определенных на предыдущем шаге:

- в процедуре `ПриСозданииНаСервере` (обработчик события формы) вставить вызов по шаблону:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументов.ПриСозданииНаСервере_ФормаДокумента(ЭтотОбъект, Элемент.<ГруппаФормы>);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

Где второй параметр `Расположение` (необязательный) - указание группы формы, в которой будет размещена декорация с данными о текущем состоянии оригинала. В противном случае, без указания параметра, декорация будет размещена в нижнем правом углу формы.

- в процедуре `ОбработчикОповещения` (обработчик события формы) добавить код:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументовКлиент.ОбработчикОповещенияФормаДокумента(ИмяСобытия, ЭтотОбъект);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

- добавить вспомогательную процедуру:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
&НаКлиенте
Процедура Подключаемый_ДекорацияСостояниеОригиналаНажатие()
    УчетОригиналовПервичныхДокументовКлиент.ОткрытьМенюВыбораСостояния(ЭтотОбъект);
КонецПроцедуры
//Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

Примеры внедрения и настройки подсистемы в формах объектов см. в документах `_ДемоПеремещениеТоваров` и `_ДемоРеализацияТоваров`.

Подключение форм списков

Для возможности отображения подключаемых команд установки состояния и сведений о текущем состоянии оригинала в формах списков объектов, необходимо:

1. Форму списка подключить к подсистеме Подключаемые команды.
2. В запрос динамического списка добавить поля:
   - произвольное выражение по шаблону:
     ```
     NULL
     ```
     Имя поля: `СостояниеОригиналаПервичногоДокумента`.
   - произвольное выражение по шаблону:
     ```
     ИСТИНА
     ```
     Имя поля: `ОбщееСостояние`.
   - произвольное выражение по шаблону:
     ```
     0
     ```
     Имя поля `СостояниеОригиналПолучен`.
3. В реквизитах формы у колонок динамического списка `Ссылка`, `ОбщееСостояние` и `СостояниеОригиналаПервичногоДокумента` поставить галочку `Использовать всегда`;
4. В обработчике события `ПриСозданииНаСервере` формы вставить вызов по шаблону:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументов.ПриСозданииНаСервере_ФормаСписка(ЭтотОбъект, Элементы.<ТаблицаФормы>, Элементы.<КолонкаДинамическогоСписк
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

Где третий параметр `Расположение` (необязательный) - указание колонки динамического списка, перед которой будет размещены колонки с данными о текущем состоянии оригинала. В противном случае, без указания параметра, новые колонки будут размещены в списке последними. Пример:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументов.ПриСозданииНаСервере_ФормаСписка(ЭтотОбъект, Элементы.Список, Элементы.Комментарий);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

5. В обработчике события `ОбработчикОповещения` формы добавить код:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументовКлиент.ОбработчикОповещенияФормаСписка(ИмяСобытия, ЭтотОбъект, Элементы.<ТаблицаФормы>);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

6. В обработчике события `Выбор` таблицы формы добавить код:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументовКлиент.СписокВыбор(Поле.Имя, ЭтотОбъект, Элементы.<ТаблицаФормы>, СтандартнаяОбработка);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

7. В обработчике события `СписокПриПолученииДанныхНаСервере` таблицы формы добавить код:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
УчетОригиналовПервичныхДокументов.ПриПолученииДанныхНаСервере(Строки);
// Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

8. Добавить вспомогательные процедуры:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
&НаКлиенте
Процедура Подключаемый_ОбновитьКомандыСостоянияОригинала()

    ОбновитьКомандыСостоянияОригинала()

КонецПроцедуры
&НаСервере
Процедура ОбновитьКомандыСостоянияОригинала()

// Код из обработчика события ПриСозданииНаСервере этой формы по подключению команд

КонецПроцедуры
//Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

Пример:

```
&НаСервере
Процедура ОбновитьКомандыСостоянияОригинала()

    ПодключаемыеКоманды.ПриСозданииНаСервере(ЭтотОбъект);

КонецПроцедуры
```

9. Опционально. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю:
   - Имя: `ПодменюУстановитьНастроитьСостояниеОригинала`;
   - Заголовок: **Состояние оригинала**;
   - Вид: `Подменю`;
   - Отображение: `Картинка и текст`;
   - Картинка: `УстановитьСостояниеОригиналаПервичногоДокумента`. и группы для вывода команд установки состояния оригинала по шаблону:
   - Имя: `ГруппаУстановитьОригиналПолучен`;
   - Вид: `Подменю`;
   - Отображение: `Картинка`;
   - Картинка: `СостояниеОригиналаПервичногоДокументаОригиналПолучен`.

Примеры внедрения и настройки подсистемы в формах списков см. в документах `_ДемоПеремещениеТоваров`, `_ДемоРеализацияТоваров` и обработке `_ДемоЖурналУчетаОригиналовПервичныхДокументов`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы следует использовать роли:

№	Роли и их назначение
1.	`ДобавлениеИзменениеСостоянийОригиналовПервичныхДокументов` Настройка списка состояний жизненного цикла оригинала документа в приложении.
2.	`ИзменениеСостоянийОригиналовПервичныхДокументов` Просмотр и изменение текущих состояний оригиналов первичных документов.
3.	`ЧтениеСостоянийОригиналовПервичныхДокументов` Просмотр текущих состояний оригиналов первичных документов.

Дополнительно следует создать вспомогательные роли или использовать подходящие роли, существующие в конфигурации, для обеспечения доступа к данным.

№	Вспомогательные роли и их назначение
1.	`<ИспользованиеОбработкиЖурналУчетаПервичныхДокументов>` Работа с журналом учета состояний оригиналов первичных документов.

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Администратор	• `ПолныеПрава` (из подсистемы **Базовая функциональность**)
2.	Ответственный, который может изменять/добавлять состояния оригиналов в справочник **Состояния оригиналов первичных документов**.	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеСостоянийОригиналовПервичныхДокументов`
3.	Пользователь, которому разрешено просматривать и изменять состояния оригиналов первичных документов	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ИзменениеСостоянийОригиналовПервичныхДокументов` • `ИспользованиеОбработкиЖурналУчетаПервичныхДокументов`.
4.	Пользователь, которому разрешен только просмотр текущих состояний оригиналов первичных документов	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ЧтениеСостоянийОригиналовПервичныхДокументов`

Особые случаи внедрения подсистемы

- См. Внедрение подсистемы «Учет оригиналов первичных документов» без подсистемы «Подключаемые команды».

Использование при разработке конфигурации

Разработка рабочего места

Подсистема предполагает работу со специализированным рабочем местом - журналом, в котором должны отображаться все документы информационной базы, по которым получены/не получены подписанные оригиналы первичных документов, документы которые еще не отслеживались, а также предусмотрены различные отборы и возможность изменения состояния первичного документа.

Основные (`СостояниеОригиналаПервичногоДокумента`, `ПечатнаяФорма` и т.д.) и дополнительные сведения (как например, **Контрагент**, **Организация**, **Вх.Номер**, **НомерДокумента**, **Комментарий**, **ДатаДокумента** и т.д.) для журнала должны браться из регистра сведений `СостоянияОригиналовПервичныхДокументов` и общего реестра документов (пример см. `_ДемоРеестрДокументов`).

В качестве основы для разработки рабочего места и пример настройки можно взять в демонстрационной базе в обработке `_ДемоЖурналУчетаОригиналовПервичныхДокументов`.

После разработки рабочего места, для открытия необходимо в общем модуле `УчетОригиналовПервичныхДокументовКлиентПереопределяемый` в процедуре `ПриОткрытииФормыЖурналаУчетаОригиналов` добавить фрагмент кода:

```
ОткрытьФорму("<ФормаРабочегоМеста>");
```

Пример см. в демонстрационной базе в общем модуле `УчетОригиналовПервичныхДокументовКлиентПереопределяемый`

Также добавить роль для использования данного рабочего места (см. в разделе настройки прав доступа).

Если в конфигурацию встроена **1С:Библиотека подключаемого оборудования**, то можно предусмотреть быструю установку конечного состояния оригинала в рабочем месте или в ином произвольном журнале с помощью сканера штрих-кода.

Для этого необходимо в общем модуле `УчетОригиналовПервичныхДокументовКлиентПереопределяемый` в процедуре `ПриПодключенииСканераШтрихкода` описать механизм подключения оборудования. Пример реализации для стандартного подключения через БПО:

```
МенеджерОборудованияКлиент.НачатьПодключениеОборудованиеПриОткрытииФормы(Неопределено, ЭтаФорма, "СканерШтрихкода");
```

Далее в основной форме рабочего места подсистемы или ином произвольном журнале:

- в обработчике события `ПриОткрытии` формы добавить вызов:

  ```
  УчетОригиналовПервичныхДокументовКлиент.ПриПодключенииСканераШтрихкода(ЭтотОбъект);
  ```

- в обработчике события `ОбработчикОповещения` формы добавить код:

  ```
  // СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
  УчетОригиналовПервичныхДокументовКлиент.ОбработатьШтрихкод(Параметр,ИмяСобытия);
  // Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
  ```

Подробнее программный интерфейс подсистемы описан в соответствующем разделе главы 4. Программный интерфейс.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Центр мониторинга

Подсистема **Центр мониторинга** предназначена для сбора **обезличенной статистики** использования конфигурации, настроек конфигурации. В базовом варианте подсистема обеспечивает сбор сведений о технологических дампах платформы, количестве записей во всех таблицах информационной базы, а также значениях функциональных опций. Однако список собираемых показателей может быть расширен при внедрении конфигурации.

Настройка

В базовом виде центр мониторинга позволяет собирать следующую информацию:

- информацию о системе;
- статистику использования конфигурации;
- при наличии подсистемы **Оценка производительности** – также агрегированную информацию по ключевым операциям.

Информация о системе:

- `ИмяКомпьютера` : «F27001CA71718F89FC74E2AA42ABD5DF»,
- `ВерсияОС` : «version 6.1 Service Pack 1 (Build 7601)»,
- `ВерсияПриложения` : «8.3.6.1848»,
- `ИдентификаторКлиента` : «1d77d7a0-3025-400f-9be0-3b5655bf2edf»,
- `ОперативнаяПамять` : «7876»,
- `Процессор` : «GenuineIntel Intel64 Family 6 Model 58 Stepping 9 3300 MHz»,
- `ТипПлатформы` : «Windows x86-64»,
- `ТекущийЯзык` : «Русский»,
- `ТекущийКодЛокализации` : «ru»,
- `ТекущийЯзыкСистемы` : «ru»,
- `ТекущийРежимЗапуска` : «Управляемое приложение»,
- `ЧасовойПоясСеанса` : «Europe/Moscow».

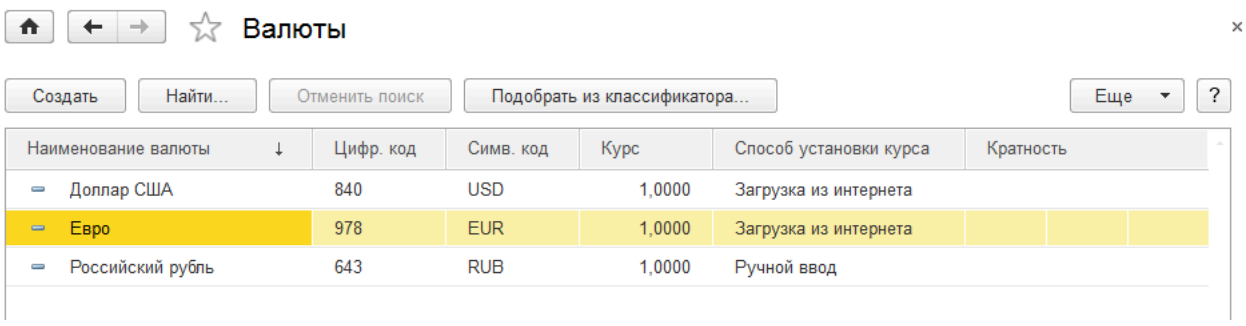
Базовая статистика по использованию конфигурации содержит количество записей в таблицах объектов метаданных:

- **План обмена**,
- **Справочник**,
- **Документ**,
- **Журнал документов**,
- **Перечисление**,
- **План видов характеристик**,
- **План счетов**,
- **План видов расчетов**,
- **Регистр сведений**,
- **Регистр накопления**,
- **Регистр бухгалтерии**,
- **Регистр расчета**,
- **Бизнес-процесс**,
- **Задача**.

Базовая статистика по настройкам конфигурации содержит значение функциональных опций с хранением значения в константе с типом `Булево` .

Если этой статистики недостаточно и необходимо, например, проанализировать заполнение данных по некоторому признаку, то надо сформировать свой текст запроса для сбора статистики в процедуре `ПриСбореПоказателейСтатистикиКонфигурации` общего модуля `ЦентрМониторингаПереопределяемый` .

Возьмем справочник `Валюты` .



```
Процедура ПриСбореПоказателейСтатистикиКонфигурации() Экспорт
    // Собрать статистику по способу установки курса справочника Валюты.
    // В регистр сведений СтатистикаКонфигураций будут внесена записи следующего вида:
    //
    // Справочник.Валюты.СпособУстановкиКурса1        Количество1
    // Справочник.Валюты.СпособУстановкиКурса2        Количество2
    // ...
    СоответствиеИменМетаданных = Новый Соответствие;

    ТекстЗапроса = "ВЫБРАТЬ
          |    СпособУстановкиКурса КАК СпособУстановкиКурса,
          |    КОЛИЧЕСТВО(*) КАК Количество
          |ИЗ
          |    Справочник.Валюты КАК Валюты
          |СГРУППИРОВАТЬ ПО
          |    СпособУстановкиКурса
          |";
    СоответствиеИменМетаданных.Вставить("Справочник.Валюты", ТекстЗапроса);

    ЦентрМониторинга.ЗаписатьСтатистикуКонфигурации(СоответствиеИменМетаданных);
КонецПроцедуры
```

> **Важно!**
>
> В целях защиты конфиденциальной информации запрещено собирать и передавать любые данные из информационной базы и сведения о ней, за исключением следующих:
>
> - хеш имени компьютера,
> - версия операционной системы,
> - версия приложения,
> - идентификатор клиента,
> - объем оперативной памяти,
> - тип центрального процессора,
> - тип платформы,
> - языковые настройки,
> - текущий режим запуска,
> - часовой пояс,
> - количество записей в таблице информационной базы,
> - количество записей в таблице информационной базы с группировкой по аналитике (группировкой аналитики могут быть только признаки, содержащие значение `Булево`, значение `Перечисления`, значение предопределенных данных).

В случае необходимости анализировать статистику информационной базы с группировкой по аналитике, содержащей конфиденциальную информацию, в качестве признака разрешается использовать хеш представления аналитики.

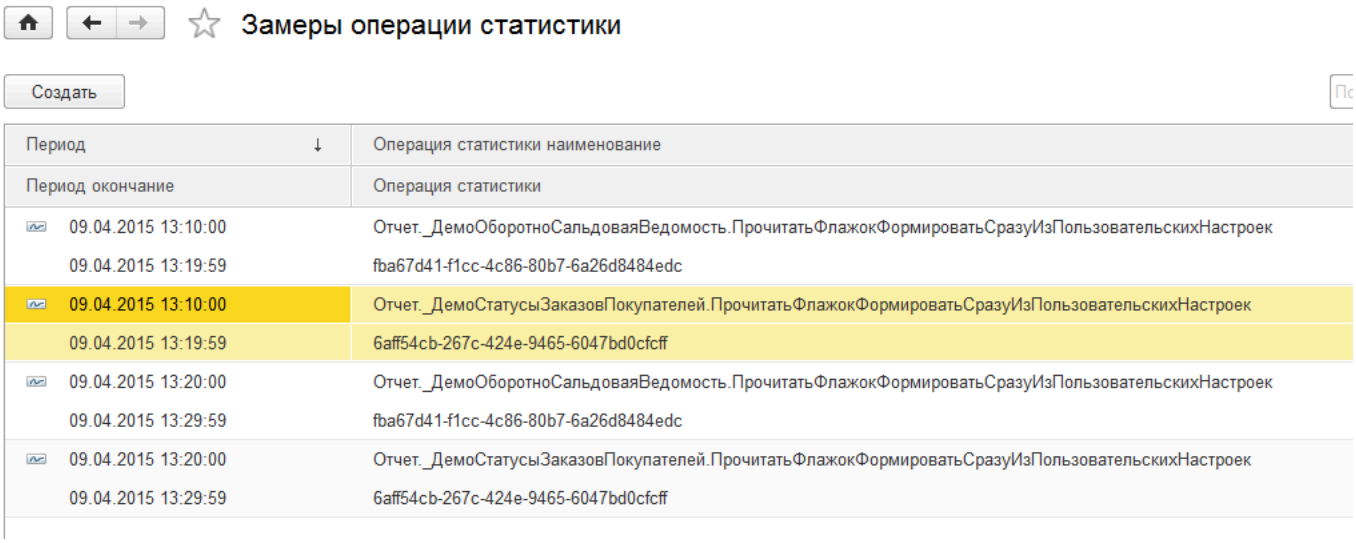
Также **Центр мониторинга** может собирать статистику использования функций конфигурации, так называемую **бизнес-статистику**. Например, необходимо анализировать информацию о том, какие отчеты и как часто используются с параметром `ФормироватьСразуИзПользовательскихНастроек`. Для регистрации данной бизнес-статистики необходимо найти место в конфигурации на сервере, где выполняется данная проверка. Необходимое место вставки программного кода – процедура `ПриСозданииНаСервере` общей формы `ФормаОтчета`.

```
Если НастройкиОтчета.ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек Тогда
    ЦентрМониторинга.ЗаписатьОперациюБизнесСтатистики(НастройкиОтчета.ПолноеИмя + ".ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек",
КонецЕсли;
```

Рекомендация: для удобного просмотра бизнес-статистики рекомендуется соблюдать порядок в именовании операций, например, `ИмяКонфигурации.ИмяПодсистемы.ИмяОперации` или `ИмяПодсистемы.ИмяОперации.СвойствоОперации`. Примеры:

- `ЦентрМониторинга.СформироватьПакетДляОтправки.Ошибка`
- `СтандартныеПодсистемы.ОбновлениеКонфигурации.Патчи.РучнаяУстановкаПатча`

После агрегации бизнес-статистики мы получим следующий результат


Замеры операции статистики

Создать

Период	Операция статистики наименование
Период окончание	Операция статистики
09.04.2015 13:10:00	Отчет._ДемоОборотноСальдоваяВедомость.ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек
09.04.2015 13:19:59	fba67d41-f1cc-4c86-80b7-6a26d8484edc
09.04.2015 13:10:00	Отчет._ДемоСтатусыЗаказовПокупателей.ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек
09.04.2015 13:19:59	6aff54cb-267c-424e-9465-6047bd0cfcff
09.04.2015 13:20:00	Отчет._ДемоОборотноСальдоваяВедомость.ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек
09.04.2015 13:29:59	fba67d41-f1cc-4c86-80b7-6a26d8484edc
09.04.2015 13:20:00	Отчет._ДемоСтатусыЗаказовПокупателей.ПрочитатьФлажокФормироватьСразуИзПользовательскихНастроек
09.04.2015 13:29:59	6aff54cb-267c-424e-9465-6047bd0cfcff

По полученным данным можно сделать следующий вывод: за период с **13:10:00 09.04.2015** по **13:29:59 09.04.2015** отчет `_ДемоОборотноСальдоваяВедомость` открывался с параметром `ФормироватьСразуИзПользовательскихНастроек` четыре раза, а отчет `_ДемоСтатусыЗаказовПокупателей` открывался с параметром `ФормироватьСразуИзПользовательскихНастроек` пять раз.

Главное отличие регистрации бизнес-статистики от статистики использования конфигурации заключается в том, что бизнес-статистика агрегирует операции. Сбор статистики использования конфигурации выполняет срез информации, предыдущие данные удаляются.

Центр мониторинга при наличии подсистемы **Оценка производительности** может анализировать и передавать TOP N худших операций производительности, влияющих на общую производительность системы (по умолчанию 10). Но в данном случае необходимо быть внимательным. Замеры оценки производительности передаются в агрегированном виде за период сбора статистики, сами замеры – не передаются. Количеством худших операций производительности управляет сервис приема информации, при необходимости собираются все агрегированные замеры за период.

Настройка прав доступа пользователей

Роли, используемые для работы с подсистемой:

№	Роли и их назначение
1.	**Администратор системы**. Принятие решения по поводу согласия/несогласия на сбор и отправку обезличенной статистики конфигурации. Настройка адреса сервиса приема данных.

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Настройка обмена данными

Объекты метаданных подсистемы не следует включать в планы обмена распределенной информационной базы (РИБ), автономного рабочего места, а также в планы обмена, предназначенные для синхронизации данных между различными приложениями, поскольку для различных узлов мониторинг настраивается и работает по различным правилам.

Шаблоны сообщений

Подсистема **Шаблоны сообщений** позволяет экономить время, создавая письма и SMS-сообщения по заранее подготовленным шаблоном. В тексте шаблонов сообщений можно определить реквизиты, которые будут заполняться данными из документов или справочников приложения. Кроме того, в шаблоне можно указать, что к письму нужно прикреплять вложения и печатные формы.

Подсистема **Шаблоны сообщений** работает совместно с подсистемами Работа с почтовыми сообщениями, Отправка SMS, а также Взаимодействия. Если какой-либо из перечисленных подсистем нет в составе конфигурации, то соответствующая функциональность шаблонов писем или SMS автоматически скрывается из интерфейса.

Настройка

Для создания шаблонов сообщений на основании документов или справочников необходимо:

1. Принять решение по поводу состава типов таких объектов (определить предметы шаблонов сообщений). Например, это могут быть документы счетов на оплату, справочники контрагентов, сотрудников и т. п.
2. Выбранные объекты нужно перечислить в свойстве `Тип` определяемого типа `ПредметШаблонаСообщения`; во всех модулях менеджеров этих объектов добавить следующие процедуры и при необходимости вписать их реализацию:

```
// СтандартныеПодсистемы.ШаблоныСообщений

// Вызывается при подготовке шаблонов сообщений и позволяет переопределить список реквизитов и вложений.
//
// Параметры:
//  Реквизиты - см. ШаблоныСообщенийПереопределяемый.ПриПодготовкеШаблонаСообщения.Реквизиты
//  Вложения  - см. ШаблоныСообщенийПереопределяемый.ПриПодготовкеШаблонаСообщения.Вложения
//  ДополнительныеПараметры - Структура - дополнительные сведения о шаблоне сообщений.
//
Процедура ПриПодготовкеШаблонаСообщения(Реквизиты, Вложения, ДополнительныеПараметры) Экспорт

КонецПроцедуры

// Вызывается в момент создания сообщений по шаблону для заполнения значений реквизитов и вложений.
//
// Параметры:
//  Сообщение - Структура:
//    * ЗначенияРеквизитов - Соответствие из КлючИЗначение - список используемых в шаблоне реквизитов:
//      ** Ключ     - Строка - имя реквизита в шаблоне;
//      ** Значение - Строка - значение заполнения в шаблоне.
//    * ЗначенияОбщихРеквизитов - Соответствие из КлючИЗначение - список используемых в шаблоне общих реквизитов:
//      ** Ключ     - Строка - имя реквизита в шаблоне;
//      ** Значение - Строка - значение заполнения в шаблоне.
//    * Вложения - Соответствие из КлючИЗначение:
//      ** Ключ     - Строка - имя вложения в шаблоне;
//      ** Значение - ДвоичныеДанные
//                  - Строка - двоичные данные или адрес во временном хранилище вложения.
//  ПредметСообщения - ЛюбаяСсылка - ссылка на объект являющийся источником данных.
//  ДополнительныеПараметры - Структура -  дополнительная информация о шаблоне сообщения.
//
Процедура ПриФормированииСообщения(Сообщение, ПредметСообщения, ДополнительныеПараметры) Экспорт

КонецПроцедуры

// Заполняет список получателей SMS при отправке сообщения сформированного по шаблону.
//
// Параметры:
//   ПолучателиSMS - ТаблицаЗначений:
//     * НомерТелефона - Строка - номер телефона, куда будет отправлено сообщение SMS;
//     * Представление - Строка - представление получателя сообщения SMS;
//     * Контакт       - Произвольный - контакт, которому принадлежит номер телефона.
//  ПредметСообщения - ЛюбаяСсылка - ссылка на объект, являющийся источником данных.
//                   - Структура  - структура описывающая параметры шаблона:
//    * Предмет               - ЛюбаяСсылка - ссылка на объект, являющийся источником данных;
//    * ВидСообщения - Строка - вид формируемого сообщения: "ЭлектроннаяПочта" или "СообщениеSMS";
//    * ПроизвольныеПараметры - Соответствие - заполненный список произвольных параметров;
//    * ОтправитьСразу - Булево - признак мгновенной отправки;
//    * ПараметрыСообщения - Структура - дополнительные параметры сообщения.
//
Процедура ПриЗаполненииТелефоновПолучателейВСообщении(ПолучателиSMS, ПредметСообщения) Экспорт

КонецПроцедуры

// Заполняет список получателей почты при отправке сообщения сформированного по шаблону.
//
// Параметры:
//   ПолучателиПисьма - ТаблицаЗначений - список получается письма:
//     * ВариантОтправки - Строка - вариант отправки для получателя письма: Кому, Копия, СкрытаяКопия, ОбратныйАдрес;
//     * Адрес           - Строка - адрес электронной почты получателя;
//     * Представление   - Строка - представление получателя письма;
//     * Контакт         - Произвольный - контакт, которому принадлежит адрес электронной почты.
//  ПредметСообщения - ЛюбаяСсылка - ссылка на объект, являющийся источником данных.
//                   - Структура  - структура описывающая параметры шаблона:
//    * Предмет               - ЛюбаяСсылка - ссылка на объект, являющийся источником данных;
//    * ВидСообщения - Строка - вид формируемого сообщения: "ЭлектроннаяПочта" или "СообщениеSMS";
//    * ПроизвольныеПараметры - Соответствие - заполненный список произвольных параметров;
//    * ОтправитьСразу - Булево - признак мгновенной отправки письма;
//    * ПараметрыСообщения - Структура - дополнительные параметры сообщения;
//    * ПреобразовыватьHTMLДляФорматированногоДокумента - Булево - признак преобразование HTML текста
//             сообщения содержащего картинки в тексте письма из-за особенностей вывода изображений
//             в форматированном документе;
//    * УчетнаяЗапись - СправочникСсылка.УчетныеЗаписиЭлектроннойПочты - учетная запись для отправки письма.
//
Процедура ПриЗаполненииПочтыПолучателейВСообщении(ПолучателиПисьма, ПредметСообщения) Экспорт

КонецПроцедуры

// Конец СтандартныеПодсистемы.ШаблоныСообщений
```

Если код процедур оставить незаполненными, то в шаблонах сообщений будут доступны реквизиты предмета и определенные в нем печатные формы. При наличии у предмета контактной информации список адресатов будет заполнен, в противном случае он будет пустой.

Пример реализации этих процедур см. в демонстрационной конфигурации в справочнике `_ДемоФизическиеЛица`.

3. При наличии в конфигурации подсистемы **Подключаемые команды**, команды отправки выводятся с ее помощью. Для этого в формы, в которых должны отображаться кнопки отправки, следует дополнительно внедрить подсистему **Подключаемые команды**. Для целей оптимизации производительности при открытии формы рекомендуется добавить в командную панель подменю для вывода команд отправки по шаблону:

- Имя: `ПодменюОтправить`.
- Заголовок: **Отправка**.
- Вид: `Подменю`.
- Отображение: `Картинка`.
- Картинка: `ОтправитьСообщение` (стандартная картинка).

Размещение в командном интерфейсе

Для использования подсистемы необходимо разместить в командном интерфейсе ответственного за список шаблонов сообщений справочник `ШаблоныСообщений` – для ведения списка шаблонов сообщений.

В случае если в конфигурации не используется подсистема Настройки программы, в рабочем месте администратора приложения необходимо разместить справочник `ШаблоныСообщений` и константу `ИспользоватьШаблоныСообщений` – для включения и выключения подсистемы. См. пример в форме `Органайзер` обработки `ПанельАдминистрированияБСП`.

Особые случаи внедрения подсистемы

- Внедрение подсистемы «Шаблоны сообщений» без подсистемы «Работа с файлами».

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Шаблоны сообщений** следует использовать роли, приведенные ниже.

№	Роли и их назначение
1.	`ДобавлениеИзменениеЛичныхШаблоновСообщений` Добавление и редактирование общих и персональных шаблонов сообщений.
2.	`ДобавлениеИзменениеШаблоновСообщений` Ведение общих шаблонов сообщений.
3.	`ЧтениеШаблоновСообщений` Просмотр шаблонов сообщений, имеющихся в приложении.
4.	`ПолныеПрава` (из подсистемы **Базовая функциональность**). Включение и отключение шаблонов сообщений, удаление помеченных на удаление объектов подсистемы

Пример настройки прав доступа пользователей:

№	Группа пользователей и ее функции	Состав ролей
1.	Ответственный за список шаблонов сообщений	• `БазовыеПраваБСП` (из подсистемы **Базовая функциональность**), • `ДобавлениеИзменениеШаблоновСообщений`

Использование при разработке конфигурации

По умолчанию в шаблонах сообщений доступны все реквизиты предмета сообщения, за исключением реквизитов с типом `ХранилищеЗначения` и некоторых стандартных реквизитов (`ПометкаУдаления`, `ЭтоГруппа` и др.). Также доступны поля контактной информации и доп. реквизиты и сведения. В качестве вложений к письмам доступны определенные в конфигурации печатные формы, внешние печатные формы и печатные формы из расширений. При наличии контактной информации у предмета будет заполонен список адресатов: для почтовых сообщений – из полей с видом контактной информации «электронная почта», а для сообщений SMS – с видом контактной информации «телефон».

В случаях, когда при создании шаблонов сообщений недостаточно стандартных средств, можно предусмотреть собственные реквизиты, печатные формы и вложения.

1. В процедуре `ПриПодготовкеШаблонаСообщения` определить состав используемых реквизитов и вложений:

```
НовыйРеквизит = Реквизиты.Добавить();
НовыйРеквизит.Имя = "СостояниеКонтрагента";
НовыйРеквизит.Представление = НСтр("ru = 'Состояние контрагента'");
НовыйРеквизит.Тип = Новый ОписаниеТипов("Строка");
Карта = Вложения.Добавить();
КартинкаКнопки.Имя = "КартаРасположенияОфиса'";
КартинкаКнопки.Представление = НСтр("ru = Карта расположения офиса'");
КартинкаКнопки.ТипФайла = "jpg";
```

2. В процедуре `ПриФормированииСообщения` заполнить ранее определенные реквизиты и вложения:

```
Сообщение.ЗначенияРеквизитов["СостояниеКонтрагента"] = ПроверкаКонтрагентов.СостояниеКонтрагента(ПредметСообщения.ИНН, ПредметСообщения.КПП
АдресКартинкиВоВременноеХранилище = КартаРасположенияОфис();
Сообщение.Вложения["КартаРасположенияОфиса"] = АдресКартинкиВоВременноеХранилище;
```

3. Если у предмета нет контактной информации или требуется особенный список адресатов, то список получателей письма или сообщений SMS должен быть заполнен или по контактной информации реквизита предмета, или используя собственный алгоритм. Для этого в модуле менеджера объекта, являющемся предметом шаблонов сообщений, в процедурах `ПриЗаполненииТелефоновПолучателейВСообщении` и `ПриЗаполненииПочтыПолучателейВСообщении` требуется заполнить получателей сообщения:

```
Процедура ПриЗаполненииПочтыПолучателейВСообщении(ПолучателиПисьма, ПредметСообщения) Экспорт

    ШаблоныСообщений.ЗаполнитьПолучателей(ПолучателиПисьма, ПредметСообщения, "Контрагент");
    Получатель = ПолучателиПисьма.Добавить();
    Получатель.Адрес = ПредметСообщения.ЭлектроннаяПочтаСтрокой;
    Получатель.Представление = ПредметСообщения.ЭлектроннаяПочтаПредставление;

КонецПроцедуры

Процедура ПриЗаполненииТелефоновПолучателейВСообщении(ПолучателиSMS, ПредметСообщения) Экспорт

    ШаблоныСообщений.ЗаполнитьПолучателей(ПолучателиSMS, ПредметСообщения, "Контрагент", Перечисления.ТипыКонтактнойИнформации.Телефон);

КонецПроцедуры
```

Пример использования см. в документе `_ДемоСчетНаОплатуПокупателю` и справочнике `_ДемоКонтрагенты` демонстрационной конфигурации.

В тех случаях, когда требуется определить список реквизитов, вложений или список получателей сразу для всех или нескольких предметов шаблонов сообщений, необходимо размещать код в одноименных процедурах общего модуля `ШаблоныСообщенийПереопределяемый`.

Разработка предметов шаблонов сообщений, не привязанных к объектам конфигурации

В случаях, когда необходимо создать назначение для шаблонов сообщений, которое не является объектом конфигурации, в процедуре `ПриОпределенииНастроек` общего модуля `ШаблоныСообщенийПереопределяемый` необходимо добавить код, например:

```
Предмет = Настройки.ПредметыШаблонов.Добавить();
Предмет.Имя = "ОповещениеКлиентаИзменениеЗаказа";
Предмет.Представление = НСтр("ru = 'Оповещение клиента ""Изменение заказа""'");
```

Затем для возможности выбора нужного шаблона в форме создать реквизит с типом `СправочникСсылка.ШаблоныСообщений` и в свойстве `ПараметрыВыбора` элемента формы (поля ввода) связанного с реквизитом указать параметры отбора:

- `Отбор.Назначение` — `Строка`, имя или представление созданного назначения;
- `Отбор.ПредназначенДляЭлектронныхПисем` или `Отбор.ПредназначенДляSMS` — `Булево`, в зависимости от необходимого вида шаблона сообщения.

Пример использования таких назначений см. в документе `_ДемоЗаказПокупателя` демонстрационной конфигурации.

Добавление общих реквизитов предметов шаблонов сообщений

Помимо реквизитов, специфичных для конкретного предмета шаблона сообщений, предусмотрена возможность определять реквизиты, общие для всех предметов. Например, для того чтобы сделать доступным для выбора реквизит **Основная организация** во всех шаблонах сообщений, необходимо описать общий реквизит в общем модуле `ШаблоныСообщенийПереопределяемый` в процедуре `ПриОпределенииНастроек`:

```
НовыйРеквизит = Настройки.ОбщиеРеквизиты.Строки.Добавить();
НовыйРеквизит.Имя = "ОсновнаяОрганизация";
НовыйРеквизит.Представление = НСтр("ru = 'Основная организация'");
НовыйРеквизит.Тип = Тип("СправочникСсылка._ДемоОрганизации");
```

Заполнение общих реквизитов производится в общем модуле `ШаблоныСообщенийПереопределяемый` в процедуре `ПриФормированииСообщения` аналогично другим реквизитам.

Программная отправка писем и SMS-сообщений по шаблонам с дополнительным отбором

Для интерактивной отправки писем и SMS-сообщений по шаблонам предназначена процедура `СформироватьСообщение` общего модуля `ШаблоныСообщенийКлиент`. При этом с ее помощью дополнительно можно ограничить список выбора доступных шаблонов, указав предмет и/или владельца шаблонов.

Например, если шаблоны писем должны различаться не только в зависимости от выбранной организации, но и от ее подразделения, то следует предварительно:

- включить справочник `Организации` в свойство `Тип` определяемого типа `ПредметШаблонаСообщения`;
- в свойстве `Тип` определяемого типа `ВладелецШаблонаСообщения` указать справочник `Подразделения`.

Затем для отправки письма по шаблону вызвать:

```
ПараметрыСообщения = Новый Структура(...); // параметры письма
ШаблоныСообщенийКлиент.СформироватьСообщение(Организация, "Письмо",, ПодразделениеОрганизации, ПараметрыСообщения);
```

Подробнее см. комментарий к процедуре `СформироватьСообщение` общего модуля `ШаблоныСообщенийКлиент`, а также пример в демонстрационной конфигурации в форме элемента справочника `_ДемоПроекты`.

Настройка обмена данными

Все объекты метаданных подсистемы, содержащие данные, рекомендуется включать в планы обмена распределенной информационной базы (РИБ) и автономного рабочего места.

Электронная подпись сервиса DSS

Подсистема **Электронная подпись сервиса DSS** позволяет формировать электронную подпись, проверять подпись, расшифровывать и шифровать с помощью закрытых ключей и сертификатов к ним, размещенных на серверах **КриптоПро DSS**. Также поддерживаются необходимые операции для работы с сервисами DSS: аутентификация, авторизация операций, создание запроса на сертификат, установка сертификата и прочее. Создание подписи, шифрование и расшифровка возможна в форматах CMS или XML. Более подробно о **КриптоПро DSS** см. https://dss.cryptopro.ru/.

Предусмотрена возможность использования REST API КриптоПро DSS 2.0 как версии v1, так и v2. При этом REST API v2 обладает расширенной функциональностью по сравнению с v1 и поддержкой мобильных приложений, разработанных на базе DSS SDK, таких как: **myDSS 2.0** и **DSS Client**.

Подсистема дополняет предусмотренные возможности в подсистеме **Электронная подпись** по формированию подписи, шифрованию, расшифровке и проверке подписи. При выполнении криптографических операций выполняются предусмотренные процедуры авторизации пользователя и подтверждения операций «вторым» фактором.

Настройка

Если подсистема внедряется вместе с подсистемой **Электронная подпись**, необходимо проверить, что в определяемом типе `ПрограммаЭлектроннойПодписи` включены два справочника: `УчетныеЗаписиDSS` и `ПрограммыЭлектроннойПодписиИШифрования`. Также необходимо предварительно выполнить все требования по настройке подсистемы **Электронная подпись**.

Если же подсистема внедряется без подсистемы **Электронная подпись**, то необходимо самостоятельно реализовать команды по вызову экспортных функций общего модуля `СервисКриптографииDSSКлиент` для требуемых операций (формирование электроннной подписи, проверка подписи, расшифровывка и т.п.).

В случае если в конфигурации не используется подсистема **Настройки программы** в рабочем месте администратора приложения поместить команды открытия форм справочников `УчетныеЗаписиSS` и `ЭкземплярыСервераDSS`, а также установки константы `ИспользоватьСервисDSS`.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы следует использовать следующие роли:

№	Роли и их назначение
1.	`ЧтениеДанныхСервисаDSS` (+ `БазовыеПраваБСП`) Чтение и просмотр информации о подключенных сервисах и списка учетных записей. Роль позволяет выполнять операции подписания, шифрования и прочие.
2.	`ДобавлениеИзменениеСерверовDSS` (+ `ЧтениеДанныхСервисаDSS`) Просмотр, добавление и изменение информации о подключенных сервисах DSS подписи и их настроек.
3.	`ДобавлениеИзменениеУчетныхЗаписейDSS` (+ `БазовыеПраваБСП`) Просмотр, добавление и изменение информации об учетных записях DSS и их настроек.
4.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Изменение всех доступных свойств любых объектов подсистемы. А также включение/выключение использования подсистемы Электронная подпись сервиса DSS

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Код проверки подписи реализуется по месту, т. к. зависит от прикладных характеристик подписываемого объекта. Пример реализации кода можно посмотреть в справочнике `Файлы` – в команде `ПроверитьВсе` в форме элемента.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме:

- регистра сведений `СертификатыПользователяDSS`, который содержит служебную информацию о сертификатах пользователя сервиса DSS.

Электронная подпись

Подсистема **Электронная подпись** предоставляет программный и пользовательский интерфейс для работы со средствами криптографии: электронная подпись, проверка электронной подписи, проверка сертификатов, шифрование и расшифровка.

Кроме того, подсистема позволяет оформить заявление на выпуск нового сертификата квалифицированной электронной подписи в удостоверяющем центре ООО «НПЦ «1С» (http://ca.1c.ru) с помощью удобного помощника.

Все криптографические операции выполняются с помощью стороннего программного обеспечения - криптопровайдера, установленного на компьютере или имеющегося на активном носителе информации (токене), поэтому для разработки приложений не требуется лицензия **ФСБ России** на шифрование и криптографию.

Настройка

В интерфейсе персональных настроек пользователя разместить команду открытия общей формы `НастройкиЭлектронной ПодписиИШифрования` с помощью процедуры `ОткрытьНастройкиЭлектронной ПодписиИШифрования` общего модуля `ЭлектроннаяПодписьКлиент`. См. пример в демонстрационной конфигурации в общей форме `_ДемоМоиНастройки`.

В случае если в конфигурации не используется подсистема **Настройки программы**, в рабочем месте администратора приложения необходимо разместить константы `ИспользоватьЭлектронныеПодписи`, `ИспользоватьШифрование` и команду открытия общей формы `НастройкиЭлектронной ПодписиИШифрования` с помощью процедуры `ОткрытьНастройкиЭлектронной ПодписиИШифрования` общего модуля `ЭлектроннаяПодписьКлиент`. См. пример в демонстрационной конфигурации в форме `ОбщиеНастройки` обработки `ПанельАдминистрированияБСП`.

Для работы помощника, позволяющего оформить заявление на выпуск сертификата, требуется внедрить подсистемы **Контактная информация** (ввода, проверки и подготовки к отправке контактной информации), **Адресный классификатор** (для подбора и проверки адреса), **Печать** (для печати заявления) и **Библиотека интернет-поддержки – Базовая функциональность** (для отправки заявления). Если этого не сделать, помощник будет скрыт. Также рекомендуется внедрить подсистему **Организации** для удобства заполнения заявления. Для эффективного использования помощника необходимо в общий модуль `ЗаявлениеНаСертификатПереопределяемый` вписать код в процедуры `ПриЗаполненииРеквизитовОрганизацииВЗаявленииНаСертификат`, `ПриЗаполненииРеквизитовВладельцаВЗаявленииНаСертификат`, `ПриЗаполненииРеквизитовРуководителяВЗаявленииНаСертификат`, `ПриЗаполненииРеквизитовПартнераВЗаявленииНаСертификат`. Пример можно посмотреть в демонстрационной конфигурации.

Для напоминаний о получении сертификата по заявлению и об окончании срока действия сертификата рекомендуется внедрить подсистему **Напоминания пользователей**. Указать в определяемом типе `ПредметНапоминания` ссылку, в типе `ПредметНапоминанияОбъект` объект справочника `СертификатыКлючейЭлектронной ПодписиИШифрования`.

Для отображения в списке текущих дел сертификатов с истекающим сроком действия и заявлений в работе рекомендуется внедрить подсистему **Текущие дела**.

Настройка подписываемых объектов

Необходимо принять решение по поводу объектов конфигурации ссылочного типа (справочников, документов и т. п.), которые могут быть подписаны. Указать их в определяемом типе `ПодписанныйОбъект`. В этих объектах должны быть реализованы команды по подписанию и проверке электронной подписи. Следует также определиться с местом размещения этих команд в формах объектов.

В выбранных объектах метаданных добавить реквизит согласно таблице:

Реквизит	Тип	Подсказка
`ПодписанЭП`	`Булево`	Признак того, что файл подписан электронной подписью.

Пример настройки реквизита можно посмотреть в демонстрационной конфигурации в справочнике **Файлы**.

Добавить функциональную опцию `ИспользоватьЭлектронныеПодписи<ИмяБиблиотеки>`, включив в нее реквизиты объектов метаданных, описанные выше (пример можно посмотреть в функциональной опции `_ДемоИспользоватьЭлектронныеПодписиБСП`).

Реализовать команды по подписанию и проверке электронной подписи с использованием экспортных функций общего модуля `ЭлектроннаяПодписьКлиент`. Пример реализации команд можно посмотреть в демонстрационной конфигурации – команды `Подписать` и `Проверить` в форме элемента справочника `Файлы`.

Принять решение по поводу объектов конфигурации ссылочного типа (справочников, документов и т. п.), которые могут быть зашифрованы. Указать их в определяемом типе `ПодписанныйОбъект`. В этих объектах должны быть реализованы команды по шифрованию и расшифровке. Следует также определиться с местом размещения этих команд в формах объектов.

В выбранных объектах метаданных добавить реквизиты согласно таблице:

Реквизит	Тип	Подсказка
`Зашифрован`	`Булево`	Признак того, что файл зашифрован.

Пример настройки реквизитов можно посмотреть в демонстрационной конфигурации в справочнике **Файлы**.

Добавить функциональную опцию `ИспользоватьШифрование<ИмяБиблиотеки>`, включив в нее реквизиты объектов метаданных, описанные выше (пример можно посмотреть в функциональной опции `_ДемоИспользоватьШифрованиеБСП`).

Реализовать команды по шифрованию и расшифровке с использованием экспортных функций общего модуля `ЭлектроннаяПодписьКлиент`. Пример реализации команд можно посмотреть в демонстрационной конфигурации – команды `Зашифровать` и `Расшифровать` в форме элемента справочника `Файлы`.

Вывод штампа визуализации электронной подписи в табличном документе

В подписанный электронной подписью табличный документ можно добавить отметку об электронной подписи (штамп визуализации электронной подписи), которая содержит информацию о лице, подписавшем документ, и о сертификате электронной подписи. Для этого в программном интерфейсе общего модуля реализована соответствующая функция `ШтампВизуализацииЭлектроннойПодписи` и процедура `ДобавитьШтампыВТабличныйДокумент`. В простейшем случае штампы электронной подписи добавляются в конец документа, но если данный вариант не подходит, то можно определить специальные области в макете для их вывода:

- Размер у такой области должен быть две колонки в ширину и семь строк в высоту, ширина колонок для данных строк должна быть установлена произвольная, отличная от ширины остального табличного документа.
- Имя области можно задать в стандартном формате: `ШтампЭП` + порядковый номер подписи, начиная с единицы – например, `ШтампЭП1`. В таком случае в процедуру `ДобавитьШтампыВТабличныйДокумент` можно передать массив штампов, которые будут в указанном порядке добавлены в обозначенные области.
  - Пример см. в макете `ПФ_MXL_СчетЗаказ` документа `_ДемоСчетНаОплатуПокупателю` демонстрационной конфигурации.

Если в конфигурации используется подсистема Работа с файлами, то для печати документа со штампами электронной подписи необходимо сохранить готовую печатную форму в присоединенных файлах в формате табличного документа (mxl), а затем, после подписания, выполнить команду **Печать** – **Со штампом электронной подписи** (в списке присоединенных файлов). Подписи при этом выводятся либо в конец документа, либо в специально отведенные области (если они названы в стандартном формате, описанном выше) в том порядке, в котором выполнялось подписание документа.

Настройка прав доступа пользователей

Для настройки прав доступа пользователей к данным подсистемы **Электронная подпись** следует использовать следующие роли:

№	Роли и их назначение
1.	`БазовыеПраваБСП` (из подсистемы **Базовая функциональность**) Просмотр сертификатов, приложений электронной подписи и шифрования, настроек подсистемы. Проверки подписей, сертификатов ключей электронной подписи.
2.	`ДобавлениеИзменениеПрограммЭлектроннойПодписиИШифрования` Добавление приложений электронной подписи и шифрования.
3.	`ДобавлениеИзменениеСертификатовКлючейЭлектроннойПодписиИШифрования` Добавление заявления на выпуск сертификата. Добавление сертификатов ключей электронной подписи. Управление оповещениями о необходимости продления сертификатов.
4.	`ДобавлениеИзменениеЭлектронныхПодписей` Выполнение подписания, усовершенствования подписей, изменение некоторых свойств своих сертификатов в справочнике. Управление оповещениями о необходимости продления сертификатов.
5.	`УдалениеЭлектронныхПодписей` Удаление подписей, сделанных другими подписантами.
6.	`ШифрованиеИРасшифровкаДанных` Выполнение операций шифрования и расшифровки данных.
7.	`РасшифровкаДанных` Выполнение расшифровки данных.
8.	`ПолныеПрава` (из подсистемы **Базовая функциональность**) Изменение всех доступных свойств любых сертификатов в справочнике. Изменение административных настроек подсистемы: • включение/отключение подписания/шифрования; • настройка списка приложений электронной подписи и шифрования; • включение/отключение выполнения операций на сервере; настройка и контроль автоматического усовершенствования подписей

Использование при разработке конфигурации

Программный интерфейс подсистемы описан в соответствующем разделе главы 4 Программный интерфейс.

Код проверки подписи реализуется по месту, т. к. зависит от прикладных характеристик подписываемого объекта. Пример реализации см. в демонстрационной конфигурации в форме элемента справочника `Файлы` – в команде `ПроверитьВсе`.

Настройка обмена данными

В планы обмена распределенной информационной базы (РИБ) и автономного рабочего места рекомендуется включать все объекты метаданных подсистемы, кроме:

- константы `АккредитованныеУдостоверяющиеЦентры`,
- константы `ДатаПоследнегоОбновленияКлассификатораОшибок`,
- константы `КлассификаторОшибокКриптографии`,
- константы `ПроверятьЭлектронныеПодписиНаСервере`,
- константы `СоздаватьЭлектронныеПодписиНаСервере`,
- регистра сведений `СпискиОтзываСертификатов`
- регистра сведений `ПутиКПрограммамЭлектроннойПодписиИШифрованияНаСерверахLinux`.

Особые случаи внедрения подсистем

В данном разделе описаны особые случаи внедрения подсистем. Как правило, эти случаи возникают из-за взаимосвязей некоторых подсистем друг с другом – когда для внедрения одной подсистемы требуется выполнить донастройку соседней и, наоборот, при внедрении одной подсистемы без другой требуется отменить ранее выполненную настройку в этой другой подсистеме.

На инструкции данного раздела приведены ссылки из инструкций по настройке соответствующих подсистем раздела 3. Инструкции приведены как для случая совместного, так и для случаев раздельного внедрения – в зависимости от способа поставки подсистем из библиотеки по умолчанию. Например, если подсистемы поставляются по умолчанию не связанными друг с другом, то дается инструкция по настройке связи. При необходимости «разорвать» связь между подсистемами необходимо выполнить обратные действия согласно инструкции.

Внедрение подсистемы «Банки» без подсистемы «Контактная информация»

По умолчанию подсистема Банки для указания страны банка использует возможности подсистемы Контактная информация. Поэтому при внедрении подсистемы **Банки** без подсистемы **Контактная информация** требуется:

- включить возможность внесения изменений в справочнике `КлассификаторБанков`,
- у реквизита `Страна` указать тип `Строка`, длина **100**, **Переменная**.

Внедрение подсистем «Бизнес-процессы и задачи» и «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к данным подсистемы Бизнес-процессы и задачи используются возможности подсистемы Управление доступом. При внедрении подсистемы **Бизнес-процессы и задачи** без подсистемы **Управление доступом** необходимо удалить шаблоны и тексты ограничений доступа, которые используют возможности подсистемы **Управление доступом**, в следующих ролях:

- `ДобавлениеИзменениеЗаданий`,
- `ИзменениеВыполнениеЗадач`,
- `ЧтениеЗаданий`.

Если подсистема **Управление доступом** внедряется без подсистемы **Бизнес-процессы и задачи**, то при первоначальном внедрении (и обновлении) конфигуратор выдаст предупреждение о наличии неразрешимых ссылок из-за отсутствия объектов метаданных подсистемы **Бизнес-процессы и задачи**. В этом случае в окне **Неразрешимые ссылки** необходимо нажать кнопку **Продолжить**.

Внедрение подсистемы «Варианты отчетов» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к дополнительным отчетам и обработкам в подсистеме Варианты отчетов используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Варианты отчетов** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменения:

- Для регистра сведений `НастройкиВариантовОтчетов` для прав `Чтение` и `Изменение` в ролях `БазовыеПраваБСП` и `БазовыеПраваВнешнихПользователейБСП` установить текст ограничения:

  ```
  ГДЕ (Пользователь = &АвторизованныйПользователь
      ИЛИ Вариант.Автор = &АвторизованныйПользователь)
  ```

- Для справочника `ВариантыОтчетов`:
  - для права `Чтение` в ролях `БазовыеПраваБСП`, `БазовыеПраваВнешнихПользователейБСП` и `ДобавлениеИзменениеЛичныхВариантовОтчетов` установить текст ограничения:

    ```
    ГДЕ (Пользовательский = ЛОЖЬ
        ИЛИ ТолькоДляАвтора = ЛОЖЬ
        ИЛИ Автор = &АвторизованныйПользователь)
    ```

  - для прав `Добавление` и `Изменение` в роли `ДобавлениеИзменениеЛичныхВариантовОтчетов` установить текст ограничения:

    ```
    ГДЕ Автор = &АвторизованныйПользователь
    ```

- Для справочника `ПользовательскиеНастройкиОтчетов`:
  - для прав `Чтение`, `Изменение` и `Добавление` в ролях `БазовыеПраваБСП`, `БазовыеПраваВнешнихПользователейБСП` и `ДобавлениеИзменениеЛичныхВариантовОтчетов` установить текст ограничения:

    ```
    ГДЕ Пользователь = &АвторизованныйПользователь
    ```

Внедрение подсистем «Взаимодействия» и «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к объектам подсистемы Взаимодействия используются возможности подсистемы Управление доступом.

Для реализации заполнения наборов значений доступа к документам подсистемы **Взаимодействия** предназначена процедура `ПриЗаполненииНаборовЗначенийДоступа` общего модуля `ВзаимодействияПереопределяемый`. Посмотреть пример реализации можно в демобазе.

При внедрении подсистемы **Взаимодействия** без подсистемы **Управление доступом** необходимо удалить шаблоны и тексты ограничений доступа, которые используют возможности подсистемы **Управление доступом**, в следующей роли:

- `ДобавлениеИзменениеВзаимодействий`.

При внедрении (или обновлении) подсистемы **Управление доступом** без подсистемы **Взаимодействия** конфигуратор выдаст предупреждение о наличии неразрешимых ссылок из-за отсутствия справочников и документов подсистемы **Взаимодействия**. В этом случае в окне **Неразрешимые ссылки** необходимо нажать кнопку **Продолжить**.

Внедрение подсистемы «Взаимодействия» без подсистемы «Электронная подпись»

По умолчанию подсистема Электронная подпись расширяет поведение подсистемы Взаимодействия. Поэтому при внедрении подсистемы **Взаимодействия** без подсистемы **Электронная подпись** требуется:

- включить возможность внесения изменений в каждый из справочников:
  - `ВстречаПрисоединенныеФайлы`;
  - `ЗапланированноеВзаимодействиеПрисоединенныеФайлы`;
  - `СообщениеSMSПрисоединенныеФайлы`;
  - `ТелефонныйЗвонокПрисоединенныеФайлы`;
  - `ЭлектронноеПисьмоВходящееПрисоединенныеФайлы`;
  - `ЭлектронноеПисьмоИсходящееПрисоединенныеФайлы`;
- в перечисленных выше справочниках:
  - удалить реквизиты `Зашифрован` и `ПодписанЭП`;
  - удалить табличные части `УдалитьЭлектронныеПодписи` и `УдалитьСертификатыШифрования`.

Внедрение подсистемы «Взаимодействия» без подсистемы «Свойства»

По умолчанию подсистема Свойства расширяет поведение подсистемы Взаимодействия. Поэтому при внедрении подсистемы **Взаимодействия** без подсистемы **Свойства** требуется:

- включить возможность внесения изменений в документ `Встреча`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в документ `ЗапланированноеВзаимодействие`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в документ `СообщениеSMS`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в документ `ТелефонныйЗвонок`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в документ `ЭлектронноеПисьмоВходящее`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в документ `ЭлектронноеПисьмоИсходящее`:
  - удалить табличную часть `ДополнительныеРеквизиты`.

Внедрение подсистемы «Дополнительные отчеты и обработки» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к дополнительным отчетам и обработкам в подсистеме Дополнительные отчеты и обработки используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Дополнительные отчеты и обработки** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменение:

- в роли `ЧтениеДополнительныхОтчетовИОбработок` удалить шаблон и текст ограничений доступа к справочнику `ДополнительныеОтчетыИОбработки`.

Внедрение подсистемы «Заметки пользователя» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к дополнительным отчетам и обработкам в подсистеме Заметки пользователя используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Заметки пользователя** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменения:

- для справочника `Заметки` для прав `Чтение` и `Изменение` в роли `ДобавлениеИзменениеЗаметок` установить текст ограничения:

```
ГДЕ Автор = &АвторизованныйПользователь
```

Внедрение подсистем «Контроль ведения учета» и «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к данным подсистемы Контроль ведения учета используются возможности подсистемы **Управление доступом**. При внедрении подсистемы **Контроль ведения учета** без подсистемы Управление доступом необходимо внести изменения в конфигурацию:

- удалить регистр сведений `КлючиДоступаКРегиструРезультатыПроверкиУчета`;
- удалить шаблоны и тексты ограничений доступа, которые используют возможности подсистемы **Управление доступом**, в роли `ЧтениеРезультатовПроверкиУчета`.

Если подсистема **Управление доступом** внедряется без подсистемы **Контроль ведения учета**, то при первоначальном внедрении (и обновлении) конфигуратор может выдавать предупреждение о наличии неразрешимых ссылок из-за отсутствия объектов метаданных подсистемы **Контроль ведения учета**. В этом случае в окне **Неразрешимые ссылки** необходимо нажать кнопку **Продолжить**.

Внедрение подсистемы «Напоминания пользователя» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к дополнительным отчетам и обработкам в подсистеме Напоминания пользователя используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Напоминания пользователя** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменения:

- для регистра сведений `НапоминанияПользователя` для прав `Чтение` и `Изменение` в роли `ДобавлениеИзменениеНапоминаний` установить текст ограничения:

```
ГДЕ Пользователь = &АвторизованныйПользователь
```

Внедрение подсистемы «Настройки программы» без подсистемы «Работа с файлами»

При внедрении подсистемы Настройки программы в конфигурацию без подсистемы Работа с файлами следует внести в конфигурацию изменение:

- в обработке `ПанельАдминистрированияБСП` удалить форму `НастройкиРаботыСФайлами`.

Внедрение подсистемы «Обмен данными» без подсистемы «Работа с почтовыми сообщениями»

По умолчанию подсистема Работа с почтовыми сообщениями расширяет поведение подсистемы Обмен данными. Поэтому на этапе внедрения подсистемы **Обмен данными** без подсистемы **Работа с почтовыми сообщениями** при выполнении операции сравнения и объединения конфигураций в окне, отображающем наличие ссылок на объекты подсистемы **Работа с почтовыми сообщениями**, требуется выбрать кнопку **Продолжить**.

Внедрение подсистемы «Пользователи» без подсистемы «Контактная информация»

По умолчанию подсистема Контактная информация расширяет поведение подсистемы Пользователи. Поэтому при внедрении подсистемы **Пользователи** без подсистемы **Контактная информация** требуется:

- включить возможность внесения изменений в справочник `Пользователи`,
- открыть **Характеристики** и удалить строку с табличной частью `КонтактнаяИнформация`,
- удалить табличную часть `КонтактнаяИнформация`.

Внедрение подсистемы «Пользователи» без подсистемы «Свойства»

По умолчанию подсистема Свойства расширяет поведение подсистемы Пользователи. Поэтому при внедрении подсистемы **Пользователи** без подсистемы **Свойства** требуется:

- включить возможность внесения изменений в справочник `Пользователи`:
  - открыть **Характеристики** и удалить две строки с видами характеристик справочника `НаборыДополнительныхРеквизитовИСведений`,
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в справочник `ВнешниеПользователи`:
  - открыть **Характеристики** и удалить две строки с видами характеристик справочника `НаборыДополнительныхРеквизитовИСведений`,
  - удалить табличную часть `ДополнительныеРеквизиты`.

Внедрение подсистемы «Пользователи» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к данным подсистемы Пользователи используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Пользователи** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменение:

- для справочника `ВнешниеПользователи` в роли `БазовыеПраваВнешнихПользователейБСП` для прав `Чтение` и `Изменение` установить текст ограничения:

```
ГДЕ Ссылка = &ТекущийВнешнийПользователь
```

- для справочника `ГруппыВнешнихПользователей` в роли `БазовыеПраваВнешнихПользователейБСП` для права `Чтение` установить текст ограничения:

```
ГДЕ Ссылка = Значение(Справочник.ГруппыВнешнихПользователей.ВсеВнешниеПользователи)
```

- для справочника `Пользователи` в роли `БазовыеПраваБСП` для права `Изменение` установить текст ограничения:

```
ГДЕ Ссылка = &ТекущийПользователь
```

Внедрение подсистемы «Работа с почтовыми сообщениями» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к данным подсистемы Работа с почтовыми сообщениями используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Работа с почтовыми сообщениями** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменение:

- для справочника `УчетныеЗаписиЭлектроннойПочты` в роли `ДобавлениеИзменениеУчетныхЗаписейЭлектроннойПочты`:
  - для права `Чтение` установить текст ограничения:

```
ГДЕ ВладелецУчетнойЗаписи = ЗНАЧЕНИЕ(Справочник.Пользователи.ПустаяСсылка)
    ИЛИ ВладелецУчетнойЗаписи = &ТекущийПользователь
```

  - для прав `Изменение` и `Добавление` установить текст ограничения:

```
ГДЕ ВладелецУчетнойЗаписи = &ТекущийПользователь
```

Внедрение подсистем «Работа с файлами» и «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к объектам подсистемы Работа с файлами используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Работа с файлами** без подсистемы **Управление доступом** необходимо удалить шаблоны и тексты ограничений доступа, которые используют возможности подсистемы **Управление доступом**, в следующих ролях:

- `ДобавлениеИзменениеПапокИФайлов`,
- `НастройкаСинхронизацииФайлов`.

При внедрении (или обновлении) подсистемы **Управление доступом** без подсистемы **Работа с файлами** конфигуратор выдаст предупреждение о наличии неразрешимых ссылок из-за отсутствия справочников `ПапкиФайлов` и `Файлы`. В этом случае в окне **Неразрешимые ссылки** необходимо нажать кнопку **Продолжить**.

Внедрение подсистемы «Работа с файлами» без подсистемы «Свойства»

По умолчанию подсистема Свойства расширяет поведение подсистемы Работа с файлами. Поэтому при внедрении подсистемы **Работа с файлами** без подсистемы **Свойства** требуется:

- включить возможность внесения изменений в справочник `Файлы`:
  - удалить табличную часть `ДополнительныеРеквизиты`;
- включить возможность внесения изменений в справочник `ПапкиФайлов`:
  - удалить табличную часть `ДополнительныеРеквизиты`.

Внедрение подсистемы «Рассылка отчетов» без подсистемы «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к данным подсистемы Рассылка отчетов используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Рассылка отчетов** в конфигурацию без подсистемы **Управление доступом** следует внести в конфигурацию изменение:

- для справочника `РассылкиОтчетов` в роли `ДобавлениеИзменениеРассылокОтчетов` для права `Чтение` установить текст ограничения:

```
ГДЕ Автор = &АвторизованныйПользователь
    ИЛИ Личная = ЛОЖЬ
    ИЛИ ЭтоГруппа = ИСТИНА
    ИЛИ ВладелецУчетнойЗаписи = &АвторизованныйПользователь
```

- для прав `Изменение` и `Добавление` установить текст ограничения:

```
ГДЕ ВладелецУчетнойЗаписи = &ТекущийПользователь
```

Внедрение подсистем «Свойства» и «Управление доступом»

По умолчанию для реализации ограничения доступа на уровне записей к дополнительным сведениям в подсистеме Свойства используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Свойства** без подсистемы **Управление доступом** необходимо удалить шаблоны и тексты ограничений доступа, которые используют возможности подсистемы **Управление доступом**, в следующих ролях:

- `БазовыеПраваБСП`, `БазовыеПраваВнешнихПользователейБСП` (только для плана видов характеристик `ДополнительныеРеквизитыИСведения` и регистра сведений `ДополнительныеСведения`),
- `ИзменениеДополнительныхСведений`,
- `ЧтениеДополнительныхСведений`.

Внедрение подсистемы «Свойства» без подсистемы «Взаимодействия»

При внедрении подсистемы **Свойства** в конфигурацию без подсистемы **Взаимодействия** следует внести в конфигурацию изменение – удалить предопределенные элементы справочника **Наборы дополнительных реквизитов и сведений**:

- `Документ_Встреча`,
- `Документ_ЗапланированноеВзаимодействие`,
- `Документ_ТелефонныйЗвонок`,
- `Документ_СообщениеSMS`,
- `Документ_ЭлектронноеПисьмоВходящее`,
- `Документ_ЭлектронноеПисьмоИсходящее`.

Внедрение подсистемы «Свойства» без подсистемы «Работа с файлами»

При внедрении подсистемы **Свойства** в конфигурацию без подсистемы **Работа с файлами** следует внести в конфигурацию изменение – удалить предопределенные элементы справочника **Наборы дополнительных реквизитов и сведений**:

- `Справочник_ПапкиФайлов`,
- `Справочник_Файлы`.

Внедрение подсистемы «Шаблоны сообщений» без подсистемы «Работа с файлами»

По умолчанию подсистема **Работа с файлами** расширяет поведение подсистемы **Шаблоны сообщений**. Поэтому при внедрении подсистемы **Шаблоны сообщений** без подсистемы **Работа с файлами** требуется:

- удалить справочник `ШаблоныСообщенийПрисоединенныеФайлы`.
- удалить подписку на события `ПереопределитьПолучаемуюФормуПрисоединенногоФайлаШаблоныСообщений`

Внедрение подсистемы «Управление доступом» без необязательных подсистем

По умолчанию для реализации ограничения доступа на уровне записей в подсистемах используются возможности подсистемы Управление доступом.

При внедрении подсистемы **Управление доступом** в ролях размещаются шаблоны, необходимые для ограничения реализации ограничений доступа в различных подсистемах. При частичном внедрении подсистем в процессе сравнения-объединения из ролей автоматически вырезаются ограничения доступа, но не вырезаются шаблоны ограничения доступа, задействованные в ограничениях.

После внедрения следует выполнить обновление шаблонов во всех ролях, включая библиотечные (недостающие шаблоны будут добавлены, лишние удалены, неактуальные обновлены).

Подробнее см. в подразделе Обновление шаблонов ограничения доступа в ролях раздела Использование при разработке конфигурации внедрения подсистемы Управление доступом.

Внедрение подсистемы «Управление доступом» без подсистемы «Варианты отчетов»

По умолчанию для вывода отчетов **Анализ прав доступа** и **Права ролей** используются возможности подсистемы **Варианты отчетов**.

При внедрении подсистемы **Управление доступом** в конфигурацию без подсистемы **Варианты отчетов** следует внести в конфигурацию изменение:

- удалить отчеты `АнализПравДоступа` и `ПраваРолей` из метаданных конфигурации.

Внедрение подсистемы «Учет оригиналов первичных документов» без подсистемы «Подключаемые команды».

По умолчанию подсистема Учет оригиналов первичных документов для вывода команд на формы списков использует возможности подсистемы **Подключаемые команды**. Поэтому при внедрении подсистемы **Учет оригиналов первичных документов** без подсистемы Подключаемые команды требуется:

- в формах списков, использующих возможности учета оригиналов первичных документов, добавить вместо вышеописанных, вспомогательные процедуры в следующем виде:

```
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
&НаКлиенте
Процедура Подключаемый_УстановитьСостояниеОригинала(Команда)

    УчетОригиналовПервичныхДокументовКлиент.УстановитьСостояниеОригинала(Команда.Имя, ЭтотОбъект,
Элементы.<ТаблицаФормы>)
КонецПроцедуры
// СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
&НаКлиенте
Процедура Подключаемый_ОбновитьКомандыСостоянияОригинала()

    ОбновитьКомандыСостоянияОригинала()

КонецПроцедуры
&НаСервере
Процедура ОбновитьКомандыСостоянияОригинала()

    УчетОригиналовПервичныхДокументов.ОбновитьКомандыСостоянияОригинала(ЭтотОбъект,Элементы.<ТаблицаФормы>);

КонецПроцедуры
//Конец СтандартныеПодсистемы.УчетОригиналовПервичныхДокументов
```

Особенности разработки объектов в расширениях конфигурации

При разработке расширений конфигурации возможно подключать собственные объекты расширений конфигурации к следующим подсистемам библиотеки:

- Варианты отчетов
- Печать
- Подключаемые команды

Подключение собственных объектов расширений конфигурации к остальным интегрируемым подсистемам недоступно или предоставляется в ограниченном виде. Это вызвано тем, что в механизме расширения конфигурации платформы **1С:Предприятие** не поддерживается создание подписок на события, бизнес-процессов и задач, расширение определяемых типов, а также расширение типа `ЛюбаяСсылка` и аналогичных. Подробнее см. раздел Особенности и ограничения в документации к механизму расширения конфигурации платформы **1С:Предприятие**. Состав этих ограничений не постоянный и может быть пересмотрен в будущих версиях платформы.

Таким образом, например, для плана обмена из расширения конфигурации невозможно задействовать возможности подсистемы **Обмен данными**, для документа или справочника - настроить ограничения доступа с помощью подсистемы **Управление доступом**, подключить к **Версионированию объектов** и т.п. Для того чтобы снять эти ограничения, рекомендуется добавлять новые объекты не с помощью расширений, а в конфигурации.

% Приложение 1. Формат файла сообщения обмена данными