

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Киев |  |  |  | kamala-soft.com |

Принципы и приемы разработки

Для разработчиков и консультантов

# Основные положения

* Стандарты разработки кода 1с доступны официально на сайте [its.1c.ru](https://its.1c.ru/db/v8std) (доступ без регистрации) и **должны быть изучены в первую очередь**.
* Принципы не противоречат стандартам, а расширяют их.

# Цели

* Облегчение будущих обновлений
* Ознакомится с полезными приемами для решения распространённых задач
* Упростить поддержку доработок другими разработчиками, и вливание новых членов команды

# Принципы:

1. Разработку всегда базируем на минимальных изменениях исходной конфигурации поставщика, избегаем «дважды изменённых» объектов как только это возможно.  
   То есть, всегда выбираем те способы решения задач, которые обеспечат более простое обновление конфигурации в будущем, даже если они несколько сложнее в реализации.
2. Обязательное комментирование кода и выделение добавленных объектов префиксами.
3. Использование общих модулей: для неоднократно используемых процедур и функция, обработчиков подписок и регламентных заданий разработчикам следует добавлять собственные модули, оставляя типовые модули неизменными.  
   Имена аналогичны именам соответствующих типовых модулей (кс\_ОбщегоНазначенияКлиент, кс\_РегламентныеЗаданияСервер и т.п.)
4. Обязательное использование подписок (их структура строго регламентирована данным документом далее).
5. Редактирование типовых форм только программным методом.
6. Если возможно, типовые роли всегда оставляем неизменными.
7. Отчеты и обработки реализуем как внешние, если не требуется иного по условиям задачи. В конфигурациях на основе БСП 2 (ERP, УТ 11, БП 3.0, ЗУП 3.0 и т. д) этот механизм значительно расширен. С его помощью без изменения конфигурации возможно создавать внешние отчеты и обработки (с размещением команды запуска в командном интерфейсе и возможностью настройки доступа различным пользователям), обработки заполнения документа, обработки создания документа на основании, дополнительные печатные формы и др.

## Дополнительные ссылки

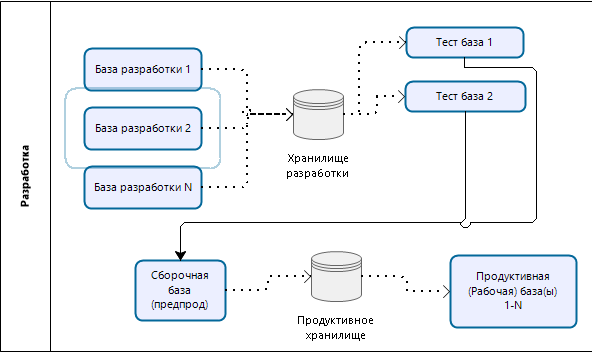
[Принципы разработки компании](https://github.com/PavelVir/Kamala_DOCs_GIT/blob/master/DOCs/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%8B%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D0%B8%20%D0%9A%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0.docx) - актуальная версия документа

[GIT Files](https://github.com/PavelVir/Kamala_DOCs_GIT/tree/master/DOCs/Files) файлы с примерами, указанными в документе.

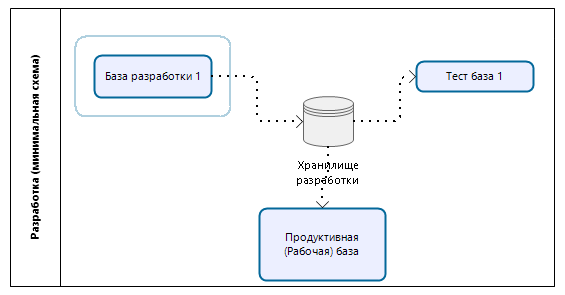
# Приемы Разработки

## Типовая схема ведения разработки

Для каждого проекта разрабатывается регламент ведения разработки и доводится всем участникам администратором проекта. Разработка обязательно должна вестись с использованием хранилища конфигурации. В некоторых случаях хранилищ конфигураций может быть несколько.



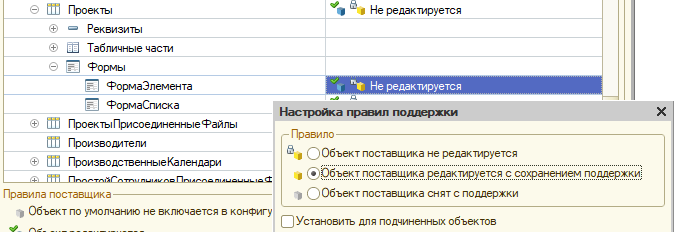
Если проект небольшой, или включает только исправления (поддержку) – может использоваться упрощенная схема, такая же схема может быть при выделении ветки разработки в отдельное хранилище.



## Работа с хранилищем (что нужно помнить)

1. Не оставляйте корень конфигурации захваченным в хранилище. При добавлении нового объекта нужно захватить корень, добавить новый объект (и минимально необходимое количество реквизитов, поскольку, например, нельзя будет обновить конфигурацию базы данных при наличии регистра накопления без регистраторов, при наличии регистра накопления без ресурсов), поместить корень и новый объект в хранилище. После этого новый объект можно заново захватить в хранилище, не блокируя корневой элемент, чтобы не мешать другим разработчикам. Если же поместить пустой регистр в хранилище, то другой разработчик после получения изменений из хранилища не сможет обновить конфигурацию базы данных.
2. При начале работы с хранилищем после открытия ее в конфигураторе выполняем «Обновить конфигурацию из хранилища». При этом в текущую базу загружаются наработки других программистов за предыдущий день, и у нас всегда будет актуальная версия конфигурации.
3. При добавлении новых объектов для ролей, у которых установлен флаг «Устанавливать права для новых объектов» (в типовых конфигурациях это ПолныеПрава), автоматически проставляются разрешительные права на новые объекты, даже если сама роль не захвачена в хранилище.
4. Если отладочные базы разработчиков создаются из копий рабочей базы, установите для каждой из них собственный заголовок главного окна, например, «Копия Иванов». Это позволит избежать ненужной путаницы при наличии нескольких одновременно открытых баз. (см. раздел Самоидентификация баз выше)
5. Каждая тестовая база разработчика подключается к хранилищу под определенным пользователем. Открыв информационную базу в конфигураторе, подключиться к хранилищу под другим пользователем не получится. Подключать новую базу под именем пользователя, для которого уже есть подключенная конфигурация, не следует, потому что это приведет к невозможности соединения предыдущей базы с хранилищем. Такая ситуация зачастую случается при создании тестовых баз разработчиков из копии рабочей базы.
6. Как захватить в хранилище сразу несколько объектов? Это можно сделать при помощи диалога работы с хранилищем (в меню Конфигурация – Хранилище конфигурации – Хранилище). В открывшемся дереве конфигурации при помощи мыши, стрелок на клавиатуре, кнопок Shift и Ctrl выделяем несколько объектов. После этого при помощи элементов контекстного меню или кнопок командной панели выполняются операции над выбранными объектами.
7. Обязательно комментируем помещение версий в хранилище и ставим номер задачи редмайн в метку версии.

## Настройка поддержки

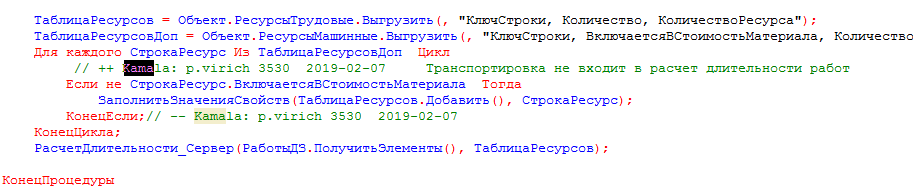
C поддержки снимаем не всю конфигурацию, а только необходимые для изменений объекты.  
Например, если нужно изменить только форму элемента – ставим на редактирование с сохранением поддержки только форму элемента.  
  
Для добавления новых объектов ставим на редактирование корень конфигурации (без подчиненных). Для снятия с поддержки конфигурации с хранилищем необходимо произвести захват корня конфигурации.

## Комментирование изменений

Комментирование используем для выделения изменений в функционале и пониманию авторства кода, для чего используем специальный шаблон, настроив под себя горячие клавиши.  
(пример шаблона в файле , ctrl+shift+t в 1с).   
Состоит из маркера начала «++», окончания «--», названия компании «Kamala», знака двоеточия «:», «ника» разработчика (как в почте) «p.virich», даты изменений, номера задачи в редмайн, и произвольного комментария (не обязателен).

**По согласованию с архитектором проекта кодировка может быть изменена под требования заказчика.**

**По некоторым проектам требуется указывать префикс проекта (например BI-group\_)**

Сначала комментируем измененный код, потом пишем свой. Если это удаление кода то будет просто комментарий старого кода.  


При добавлении реквизита или объекта добавляем префикс «кс\_», в синониме исправляем префикс на (Камала) если это новый объект метаданных, и ставим сокращенный комментарий.  
  
Если новые реквизиты, формы, макеты добавляются в объекты, промаркированные префиксом, нет необходимости указывать префикс повторно.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Новые объекты | Подчиненные для типовых объектов | Подчиненные для добавленных ранее объектов |
| Имя | кс\_ | кс\_ | - |
| Синоним | (Камала) | - | - |
| Комментарий | //Камала: p.virich 995 | //Камала: p.virich 995 | //Камала: p.virich 991 |

Модули объектов, которые добавлены нами, маркируем в начале автором.

По возможности, в пределах разумного, следует минимизировать количество кода, добавляемое в типовые модули конфигурации путем выноса части кода в наши общие модули.  
В случае «вкрапления» кода одного разработчика в код другого – используется вложенное маркирование.  
  
При добавлении процедур и функций в типовые модули обязательно размещаем закрывающий комментарий на строке конца функции/процедуры и комментируем каждую процедуру/функцию отдельно. (Это важно для по-процедурного сравнения модулей).



## Использование шаблонов

Обязательно при разработке использовать шаблоны кода, для удобства разработки (вызываются горячей клавишей Ctrl+Q)

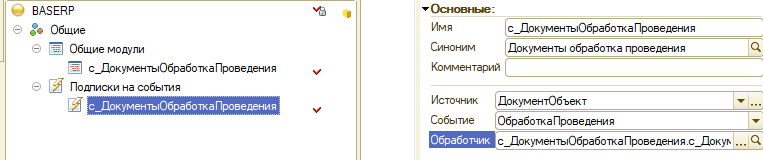
(например, , настройка через ctrl+shift+t в 1с)

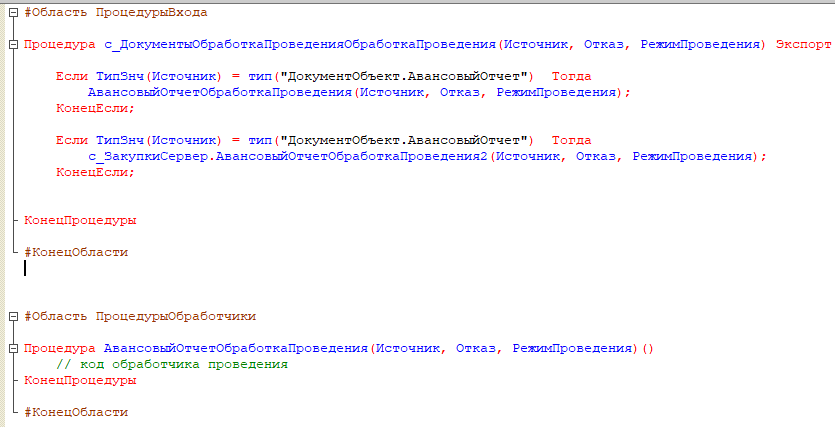
## Подписки на события

Для переопределения типовых событий используем только их.

Одно событие – одна подписка. Источник – все объекты в рамках класса. Каждая подписка – отдельный одноименный модуль.

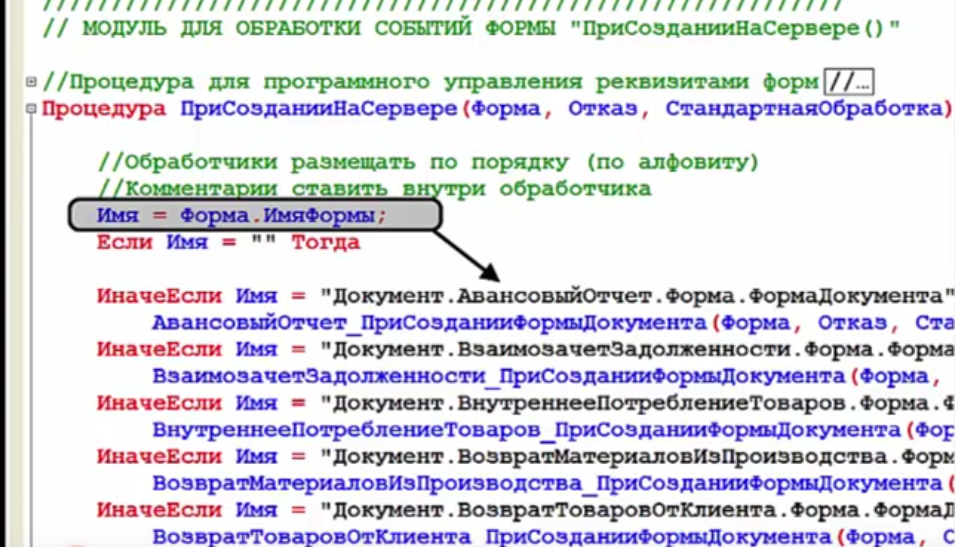
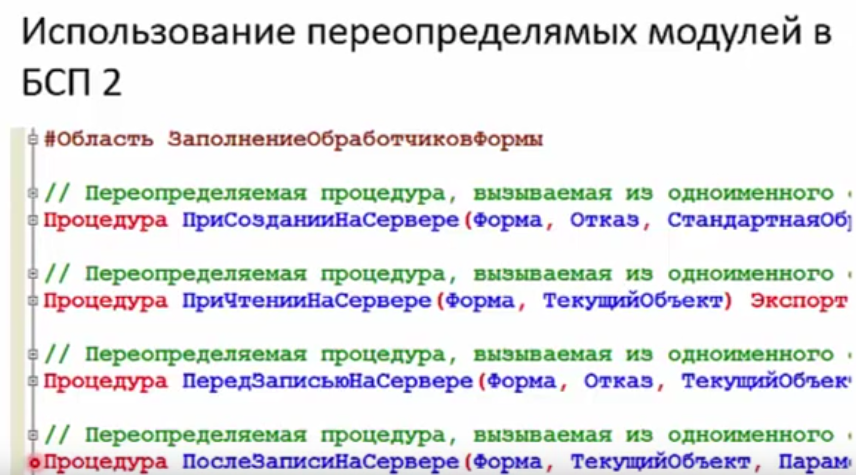
Не создаем «зоопарк» подписок, а делаем одну подписку на событие для типа источника (например, Документ Объект) и одноименный общий модуль, в котором разделяем наши обработчики по типу источника.

  
Обработчики размещаем или в этом же модуле, или (для больших подсистем) выносим в отдельные общие модули с обработчиками подсистемы.



## Модификация форм

Все изменения типовых форм поставщика производим только программным методом, размещая наши обработчики в переопределяемых модулях БСП2+ (для конфигураций на БСП общие модули с постфиксом «переопределяемый», например, «МодификацияКонфигурацииПереопределяемый. ПриСозданииНаСервере()» вызывается для всех типовых форм ) или наших общих модулях.

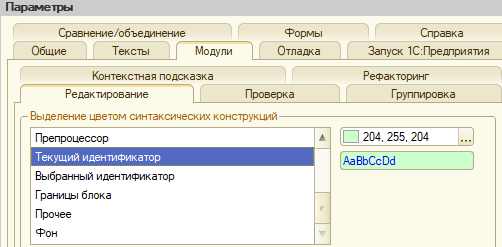
Для упрощения трудоемкости программного создания сложных форм используем средства автоматизации – де-компиляторы форм.  
(например (с) [infostart.ru/304736](https://infostart.ru/public/304736/))

## Изменения ролей

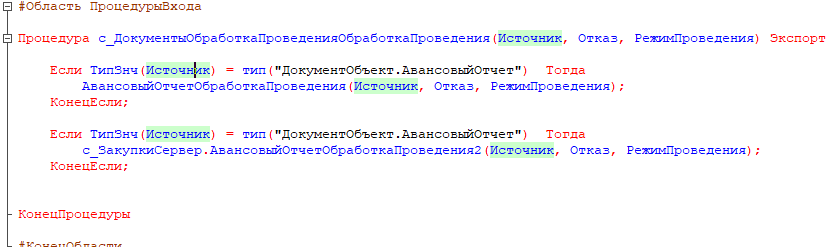
Типовые роли никогда не трогаем, для добавления/изменения функционала добавляем свои роли (Исключение: ПолныеПрава - всегда удаляем из них непосредственное удаление новых объектов).

## Выделение кода цветом

По умолчанию в конфигураторе выключено, обязательно включаем.



Позволяет часть ошибок с именами исключить уже на стадии кодирования и облегчает разработку.

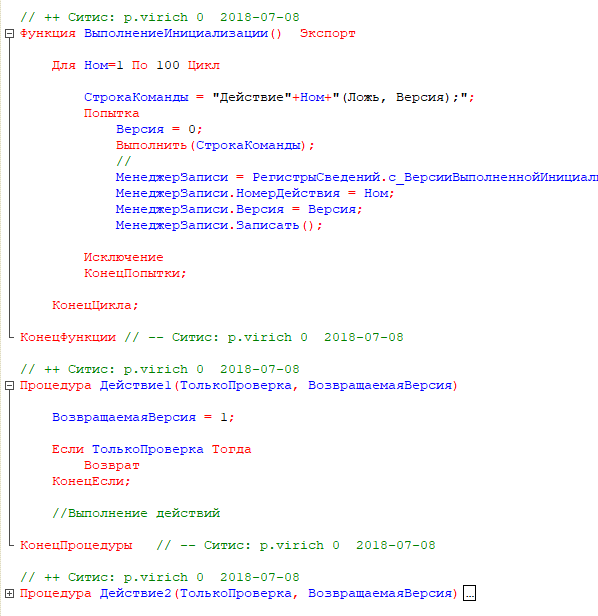
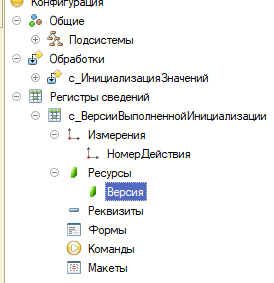


## Инициализация значений

Часто для инициализации значений необходимы механизмы, которые должны выполнится в предприятии без участия разработчика. Например, заполнение реквизитов предопределенных значений или миграция данных по реквизитам.

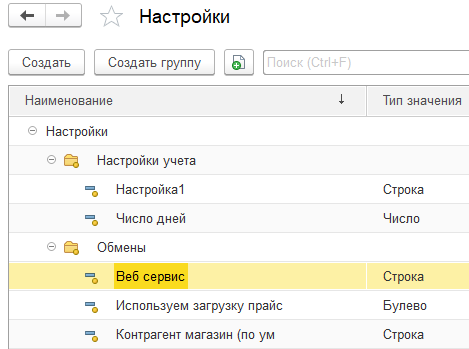
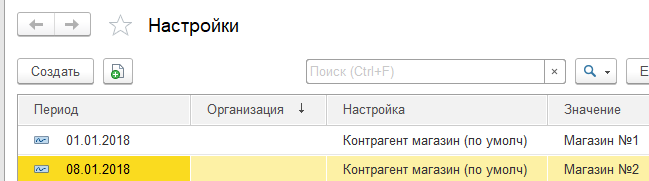
Для конфигурации на БСП следуем рекомендациям стандарта в разделе «Обработчики обновления информационной базы».

Для прочих случаев, добавляем обработку и регистр, если их еще нет. Пишем обработчики событий на свободном номере. Добавляем запуск в ПриНачалеРаботыСистемы() (например через вызов сервера из общего серверного модуля).

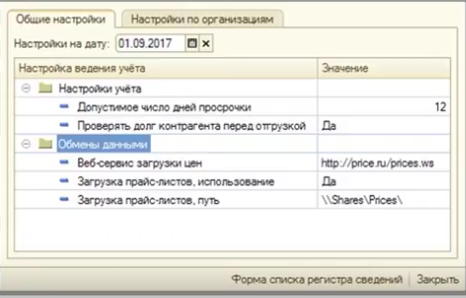
Принцип: в обработке – текущая версия, в регистре прошлая, при старте системы проверка и выполнение действий, обновление версии.  
  


## Механизм хранения настроек

Добавляем, если еще нет в конфигурации план видов характеристик, регистр сведений «кс\_Настройки», к значениям которого потом можем обращаться в коде и СКД в виде «РегистрыСведений.кс\_Настройки.ПолучитьНастройку(ПланыВидовХарактеристик.кс\_Настройки.КонтрагентМагазинПоУмолчанию)»

При необходимости можно добавить форму настроек для удобства



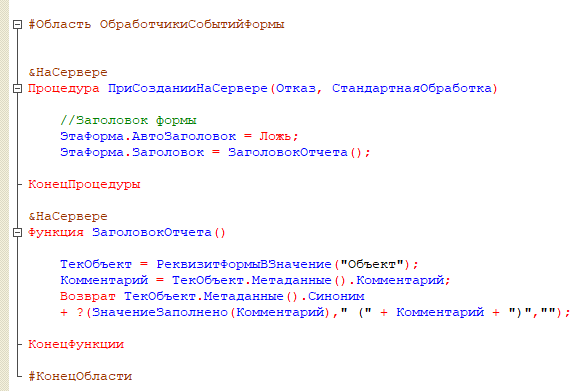
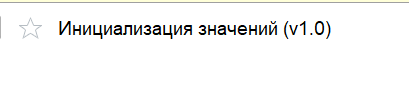
## Самоидентификация баз

В дополнение к типовому механизму БСП2 рекомендуется использовать дополнительный механизм контроля в виде:

1. Константа с адресом рабочей базы – можно через механизм настроек (см. выше)
2. Проверка при старте ИБ, рабочая ли это база – можно через механизм инициализации (см. выше)
3. Замена заголовка системы для нерабочих баз
4. В критичных регламентных заданиях и сервисах делаем проверку и логику для рабочих и тестовых баз (обмены данными, сервисы, уведомления, «тяжелые» регламентные задания)

## Версии внешних отчётов и обработок

При использовании внешних отчетов и обработок в поле «Комментарий» следует указывать строку, отображающую условную версию объекта. Например, начиная с «v1.0» и далее увеличивая старший или младший номер (в зависимости от объёма изменений) при каждом цикле доработки объекта. Данный комментарий доступен из кода, и при открытии формы его следует добавлять к заголовку окна, тогда версия обработки или отчета всегда будет видна в заголовке формы.

## Настройка печати в конфигурациях на БСП

Для подключения документа к подсистеме печати БСП

1. Модуль формы документа "Процедура ПриСозданииНаСервере" добавить вызов  
    // СтандартныеПодсистемы.Печать  
    УправлениеПечатью.ПриСозданииНаСервере(ЭтаФорма, Элементы.ПодменюПечать);  
    // Конец СтандартныеПодсистемы.Печать
2. Модуль формы документа - добавить процедуру  
   // СтандартныеПодсистемы.Печать  
   &НаКлиенте  
   Процедура Подключаемый\_ВыполнитьКомандуПечати(Команда)   
    УправлениеПечатьюКлиент.ВыполнитьПодключаемуюКомандуПечати(Команда, ЭтаФорма, Объект);   
   КонецПроцедуры  
   // Конец СтандартныеПодсистемы.Печать
3. Форма документа - добавить элемент формы группа "ПодменюПечать" в командную панель формы.
4. Модуль менеджера документа

#Область Печать

// Заполняет список команд печати.

// Параметры:

// КомандыПечати - ТаблицаЗначений - состав полей см. в функцииУправлениеПечатью.СоздатьКоллекциюКомандПечати

Процедура ДобавитьКомандыПечати(КомандыПечати) Экспорт

КонецПроцедуры

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

КонецПроцедуры

#КонецОбласти

1. Общий модуль "УправлениеПечатьюПереопределяемый" Процедура ПриОпределенииОбъектовСКомандамиПечати  
   добавить по аналогии нужные объекты

например:

СписокОбъектов.Добавить(Документы.ИНАГРО\_КорректировкаЗатратОперативногоУчета);

1. Форма списка документа - добавить элемент формы группа "ПодменюПечать" в командную панель формы.
2. Модуль формы списка документа "Процедура ПриСозданииНаСервере" добавить вызов   
   // СтандартныеПодсистемы.Печать  
    УправлениеПечатью.ПриСозданииНаСервере(ЭтаФорма, Элементы.ПодменюПечать);   
   // Конец СтандартныеПодсистемы.Печать
3. Модуль формы списка документа - добавить процедуру  
   // СтандартныеПодсистемы.Печать  
   &НаКлиенте   
   Процедура Подключаемый\_ВыполнитьКомандуПечати(Команда)   
    УправлениеПечатьюКлиент.ВыполнитьПодключаемуюКомандуПечати(Команда, ЭтаФорма, Элементы.Список); КонецПроцедуры  
   // Конец СтандартныеПодсистемы.Печать