Mobile SMARTS 3.0: Работа с компонентой

Введение

Для организации обмена данными между учетной системой пользователя и Mobile SMARTS платформы версии 3.0 применяются две специальные COM-компоненты. Одна из них универсальная, вторая предназначена специально для работы с «1С:Предприятием». Установка и регистрация компонент выполняется автоматически при установке Mobile SMARTS из дистрибутива.

Cleverence.Warehouse.StorageConnector (файл компоненты **Cleverence.MobileSMARTS.ComConnector.dll**) содержит методы для работы с Mobile SMARTS из любой программы или информационной системы, которая способна использовать COM.

AddIn.Cl.TerminalConnector (файл компоненты **Cleverence.Warehouse.TerminalConnector.dll**) предназначена специально для работы с «1С:Предприятием» и может быть подключена в 1С как внешняя компонента. Отличие от Cleverence.Warehouse.StorageConnector состоит в том, что функции AddIn.Cl.TerminalConnector «понимают» и принимают объекты 1С (массивы, списки значений, таблицы значений и др.). Благодаря этому написание кода 1С для обмена существенно облегчается.

Файлы, необходимые для работы компонент, находятся после установки по умолчанию в папке **C:\Program Files(x86)\Cleverence Soft\Mobile SMARTS\Connectivity.**

В этой же папке находится файл "**Зарегистрировать COM.bat**", запустив который (с правами Администратора) можно зарегистрировать компоненты. Это может потребоваться, если автоматическая регистрация при установке из дистрибутива не выполнилась по каким-то причинам или регистрация стала недействительной (например, из-за повреждения реестра Windows).

После установки компоненты становится возможным создание и операции с объектами компоненты непосредственно в процедурах учетной системы.

Операция создания объектов, описанных в COM, специфична для каждой системы:

Псевдокод:

```
var connector = new Cleverence.Warehouse.StorageConnector();
```

C# (подключив Cleverence.MobileSMARTS.ComConnector.dll в References):

```
var connector = new Cleverence.Warehouse.StorageConnector();
```

«1С:Предприятие 7»:

```
connector = СоздатьОбъект("Cleverence.Warehouse.StorageConnector");
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector");
```

Microsoft Dynamics AX (Axapta):

```
var connector = new COM("Cleverence.Warehouse.StorageConnector");
```

Если при попытке создания COM-объекта в Вашей учетной системе возникает ошибка, запустите (с правами Администратора) файл "**Зарегистрировать COM.bat**". Если регистрация выполнилась успешно,

Вы увидите надпись в окне командной строки: “Типы зарегистрированы успешно”. Если и после этого создание компоненты происходит с ошибкой, проверьте есть ли у кода, выполняющегося в учетной системе права на работу с COM-объектами.

Ниже указаны наиболее важные средства взаимодействия с сервером Mobile SMARTS и классы бизнес-сущностей системы.

Обмен с базой данных и инфраструктура

Класс	Описание
StorageConnector [Соединение]	Класс, объекты которого служат для доступа к базе данных Mobile SMARTS. После создания объекта требуется установить соединение с базой данных Mobile SMARTS с помощью функции SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС] после чего будет возможен обмен данными (справочниками, документами).
AppInstance [БазаСМАРТС]	Объект, содержащий описание базы данных Mobile SMARTS. База данных представляет собой хранилище конфигурации Mobile SMARTS (Environment [Среда]), текущих данных (справочников, документов, списка пользователей), а также настроек. Данные базы хранятся в папке базы на диске компьютера. Каждая база данных является экземпляром некоторого приложения Mobile SMARTS (App [ПриложениеСМАРТС]). Данный объект содержит описание базы данных (идентификатор, наименование, способ подключения), позволяет получить доступ к пользовательским настройкам, но не содержит данные базы (конфигурацию, справочники, документы). Для доступа к данным базы используется StorageConnector [Соединение].
App [ПриложениеСМАРТС]	Объект приложения Mobile SMARTS. Содержит описание некоторого приложения (идентификатор, наименование и др.). Приложение представляет собой определенное прикладное решение. Примеры приложений: "Магазин 15", "Учет имущества", "Пустая конфигурация Mobile SMARTS" (используется для разработки с нуля своего решения).

Главные бизнес сущности

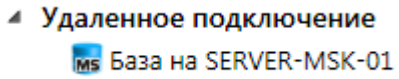
Класс	Описание
Product [Товар]	Позиция номенклатуры (товар).
Packing [Упаковка]	Упаковка товара. Каждый товар должен содержать как минимум одну упаковку.
Document [Документ]	Документ. Основная единица взаимодействия

	между учетной системой и пользователем. Содержит описание задания, назначенного к выполнению, и, собственно результат выполнения задания.
DocumentItem [СтрокаДокумента]	Строка в документе, заявленная или фактическая. Содержит информацию о товаре, его количестве и дополнительных полях (сроки годности, серийные номера, дополнительные характеристики и т.п.)

Приложения и базы

Приложение Mobile SMARTS (см. App [ПриложениеСМАРТС]) представляет собой некоторое прикладное решение на платформе Mobile SMARTS ("Магазин 15", "Учет имущества", и т.д.).
Каждое приложение устанавливается из отдельного дистрибутива (если при этом платформа Mobile SMARTS на компьютере не установлена, сначала будет предложено загрузить и установить платформу). Например, с этой страницы можно скачать продукт Mobile SMARTS для ЕГАИС, предназначенный для учета алкогольной продукции при помощи терминалов сбора данных.
После установки приложения появляется возможность создавать базы данных на основе шаблона установленного приложения. См. подробнее.
База данных Mobile SMARTS (см. AppInstance [БазаСМАРТС]) - это экземпляр определенного приложения. База данных хранит конфигурацию Mobile SMARTS и текущие данные (справочники, документы, настройки). База данных находится в определенной папке на диске компьютера. В зависимости от того, какой способ обмена данными с терминалами требуется, база данных может работать в следующих режимах: обмен с сервером Mobile SMARTS, прямое подключение терминала к компьютеру через провод,
обмен через папку на диске (Режим RDP - общая папка, терминал подключается физически к другому компьютеру, обмен непосредственно с терминалом выполняет специальная утилита, установленная на компьютере, к которому подключается терминал).
См. подробнее.

После установки Mobile SMARTS пользователь имеет возможность работать с базами данных (добавлять базы в список зарегистрированных баз, удалять, изменять настройки) с помощью Менеджера баз. Обмен данными возможен только с базами, зарегистрированными в списке баз на данном компьютере.
Удаленная база. Существует возможность добавить в список (зарегистрировать) не только базу, находящуюся на данном компьютере, но и работающую на сервере Mobile SMARTS, расположенном на другом компьютере в сети. В менеджере баз такая база будет находится в разделе Удаленное подключение:



На локальном компьютере удаленная база также, как и локальная, будет находится в заданной при добавлении базы папке, но в этой папке будут содержаться только настройки соединения с удаленным сервером Mobile SMARTS и пользовательские настройки, все данные (справочники, документы) при обмене будут загружаться (и выгружаться) с удаленного сервера. Сервер Mobile SMARTS представляет собой web-сервис, при обмене данными компонента вызывает функции web-сервиса, данные передаются по протоколу http. Организовывать общий доступ к папке базы на удаленном компьютере не нужно.
Объектам, служащим описанием удаленной базы, является также AppInstance [БазаСМАРТС] и для доступа к данным используется StorageConnector [Соединение]. При использовании функций обмена нет никакой разницы, удаленная база или локальная.

Создание и настройка новой базы данных

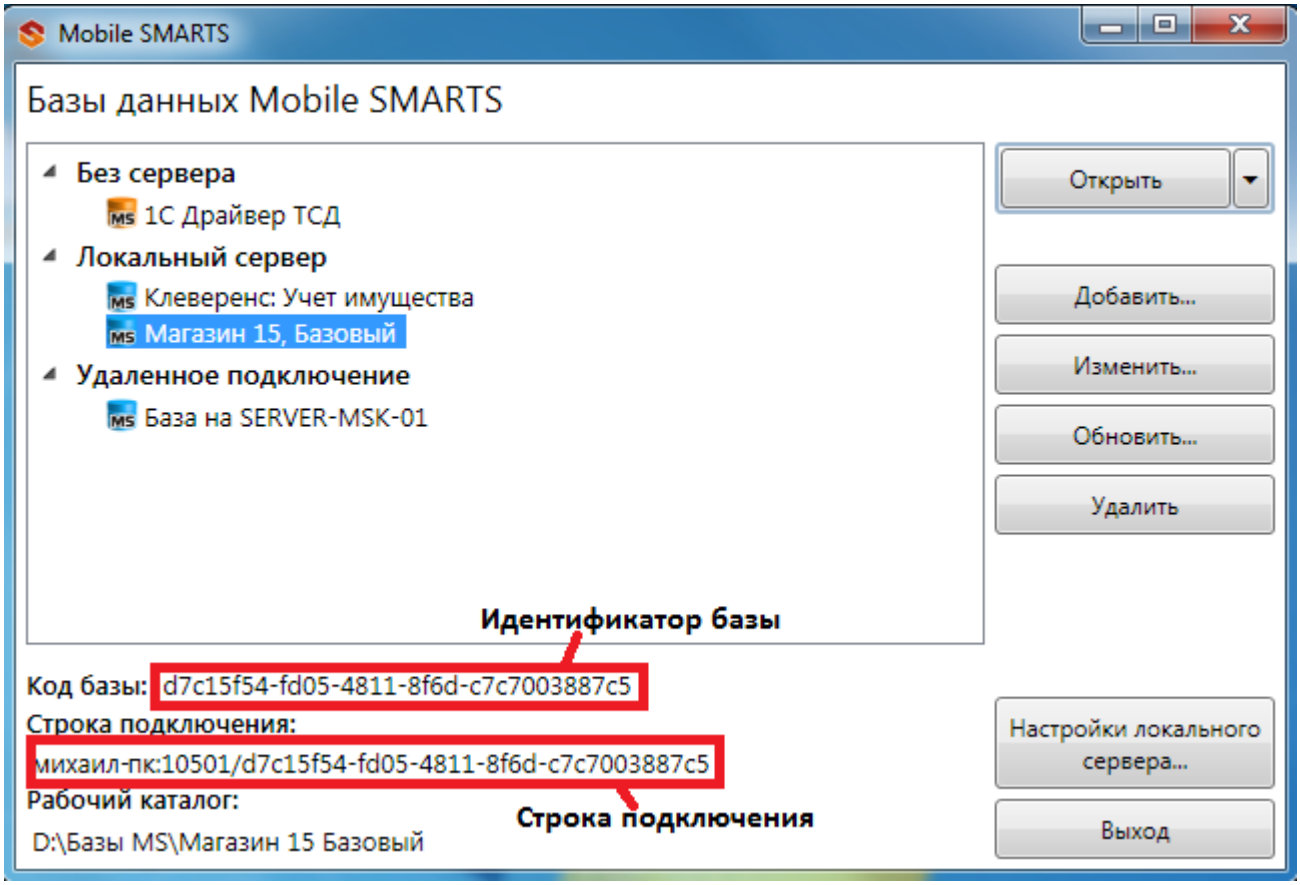
См. Заведение базы данных с помощью менеджера баз Mobile SMARTS

Для созданной базы данных пользователь указывает нужный режим работы (Прямое подключение к устройству, Прямая работа с папкой (RDP режим), Подключение к серверу). См. Настройка базы данных Mobile SMARTS.

Подключение к базе данных из учетной системы

Для того, чтобы была возможность производить обмен данными с базой Mobile SMARTS, необходимо сначала выполнить подключение к базе. Нам потребуется создать экземпляр COM-объекта Cleverence.Warehouse.StorageConnector и вызвать функцию SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС], передав в качестве аргумента **Идентификатор (Код)** базы или **Строку подключения** к базе.

Идентификатор и код в менеджере баз:



C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта
try
{
  // Выполняем подключение
  connector.SelectCurrentApp("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
}
catch
{
  // При подключении возникла ошибка, обрабатываем исключение
}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
Попытка
  connector.УстановитьПодключениеСБазойСМАРТС("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
//Выполняем подключение
Исключение
// При подключении возникла ошибка, обрабатываем исключение
КонецПопытки;
```

Получение списка баз, зарегистрированных на компьютере

Получить список баз может быть нужно, например, для того, чтобы дать пользователю учетной системы возможность выбирать базу, с которой будет выполняться обмен, из списка баз и хранить это как настройку в учетной системе, вместо того, чтобы жестко прописывать в коде строку подключения.
Воспользуемся для этого функцией GetAppsList [ПолучитьСписокБаз]. Перед вызовом этой функции выполнять подключение к какой-либо базе не нужно. Функция возвращает как локальные, так и удаленные базы, зарегистрированные на данном компьютере.

C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта

try
{
  var appInstanceList = connector.GetAppsList(""); //Получаем список баз - объект AppInstanceCollection
  for(int i=0; i<appInstanceList.Count; i++) // Обходим в цикле полученный список
  {
      var appInstance = appInstanceList[i]; //Получаем элемент списка - объект AppInstance
      var id = appInstance.Id; //Идентификатор базы
      var connectionString = appInstance.ConnectionString; //Строка подключения
      var name = appInstance.Name; //Имя базы

      // Обрабатываем полученнные значения
  }
}
catch
{
   // Обработка исключения
}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
Попытка
 СписокБаз = connector.ПолучитьСписокБаз(“”); //Получаем список баз - объект AppInstanceCollection
 Для Инд=0 По СписокБаз.Количество - 1 Цикл // Обходим в цикле полученный список
   БазаСмартса = СписокБаз.Item(Инд); //Получаем элемент списка - объект AppInstance
   Ид = БазаСмартса.Ид; //Идентификатор базы
   СтрокаПодключения = БазаСмартса.СтрокаПодключения; //Строка подключения
   Имя = БазаСмартса.Имя; //Имя базы
   … // Обрабатываем полученнные значения
 КонецЦикла;
Исключение
// Обработка исключения
КонецПопытки;
```

Получение и сохранение произвольных настроек

Есть возможность сохранять произвольные настройки в базе данных Mobile SMARTS из учетной системы. Например, это могут быть какие-либо настройки, используемые при выгрузке номенклатуры, параметры отборов документов и др. Настройки сохраняются в виде “Ключ-Значение”, Ключ - строка, задающее имя настройки, Значение - данные простого типа (строка, число, булево). Для работы с настройками необходимо получить объект базы AppInstance [БазаСМАРТС] и использовать функции GetSettings [ПолучитьНастройки] и SaveSettings [СохранитьНастройки]. Объектом настроек является AppInstanceSettings [НастройкиБазыСМАРТС].

C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта
try
{
  // Получаем базу по строке подключения или идентификатору
  var appInstance = connector.GetAppById("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");

  var appInstanceSettings = appInstance.GetSettings(); //Получаем объект настроек AppInstanceSettings
  var settingNode = appInstanceSettings.GetSetting("ИмяНастройки"); //Получаем узел настройки по имени

  var value = settingNode.Value; //Значение настройки
  settingNode.Value = newValue; // Присваиваем новое значение настройке

  appInstance.SaveSettings(appInstanceSettings); // Сохраняем настройки в базу
}
catch
{
  // Обработка исключения
}
```

«1С:Предприятие 8»:

```
connector = новый СОМОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
Попытка
 БазаСМАРТС = connector.ПолучитьБазуСМАРТСПоИД("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
 //Получаем базу по строке подключения или идентификатору

 НастройкиБазы = БазаСМАРТС.ПолучитьНастройки(); //Получаем объект настроек AppInstanceSettings
 УзелНастройки = НастройкиБазы.Настройка("ИмяНастройки"); //Получаем узел настройки по имени

 ЗначениеНастройки = УзелНастройки.Значение; //Значение настройки
 УзелНастройки.Значение = НовоеЗначение; // Присваиваем новое значение настройке

 БазаСМАРТС.СохранитьНастройки(НастройкиБазы); // Сохраняем настройки в базу
Исключение
// Обработка исключения
КонецПопытки;
```

Получение режима работы базы

Рассмотрим, как получить, в каком режиме работает база данных (Прямое подключение к устройству, Прямая работа с папкой (RDP режим) или Подключение к серверу, см. Настройка базы данных Mobile SMARTS). Нам нужно будет получить объект базы AppInstance [БазаСМАРТС], получить настройки базы с помощью функции GetSettings [ПолучитьНастройки] и использовать следующие свойства объекта настроек AppInstanceSettings [НастройкиБазыСМАРТС]:

Свойство	Тип	Описание
ServerMode [РаботаССервером]	Булево	Режим работы с сервером. Истина - база работает в режиме с сервером, Ложь - без сервера.
LocalServerMode [РаботаСЛокальнымСервером]	Булево	Возвращает признак, работает ли серверная база на локальном компьютере. Истина - серверная база на локальном компьютере, Ложь - удаленная серверная база или база не серверная (прямой обмен с ТСД или с папкой).
RemoteServerMode	Булево	Возвращает признак того, что база работает на

[РаботаСУдаленнымСервером]		удаленном сервере. Истина - серверная база на удаленном компьютере, Ложь - локальная серверная база или база не серверная (прямой обмен с ТСД или с папкой).
FolderMode [РаботаСКаталогом]	Булево	Возвращает признак того, что база работает в режиме "с каталогом". Истина - база работает работает в режиме "с каталогом". Режим RDP - обмен данными через общую папку, терминал подключается физически к другому компьютеру, обмен непосредственно с терминалом выполняет специальная утилита, установленная на компьютере, к которому подключается терминал.
DeviceMode [РаботаСУстройствомНапрямую]	Булево	Возвращает признак того, что база работает в режиме напрямую с терминалом (батч-режим). Истина - база работает в режиме напрямую с терминалом (батч-режим). Обмен с терминалом происходит через проводное подключение.

C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта соединения

try
{
  // Получаем базу по строке подключения или идентификатору
  var appInstance = connector.GetAppById("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");


  var appInstanceSettings = appInstance.GetSettings(); //Получаем объект настроек AppInstanceSettings
  if(appInstanceSettings.ServerMode == true)
  {
     //Работа с сервером
     if(appInstanceSettings.LocalServerMode == true)
     {
       // Локальный сервер
     }
     else
     {
       // Удаленный сервер
     }
  }
  else if(appInstanceSettings.FolderMode == true)
  {
     // Работа с папкой
  }
  else if(appInstanceSettings.DeviceMode == true)
  {
     // Работа с ТСД напрямую
  }
}
catch
{
   // Обработка исключения
}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
//соединения
Попытка
  БазаСМАРТС = connector.ПолучитьБазуСМАРТСПоИД("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
  //Получаем базу по строке подключения или идентификатору

  НастройкиБазы = БазаСМАРТС.ПолучитьНастройки(); //Получаем объект настроек AppInstanceSettings
  Если НастройкиБазы.РаботаССервером = Истина Тогда //Работа с сервером
     Если НастройкиБазы.РаботаСЛокальнымСервером = Истина Тогда // Локальный сервер
      ...
     Иначе // Удаленный сервер
      ...
     КонецЕсли;
  ИначеЕсли НастройкиБазы.РаботаСКаталогом = Истина Тогда //Работа с папкой
     …
  ИначеЕсли НастройкиБазы.РаботаСУстройствомНапрямую = Истина Тогда //Работа с ТСД напрямую
     ...
  КонецЕсли;

Исключение
// Обработка исключения
КонецПопытки;
```

Выгрузка номенклатуры

Выгрузка справочника номенклатуры является в большинстве случаев первоочередной задачей при интеграции Mobile SMARTS с внешней учетной системой.

Выгрузка необходима, чтобы иметь возможность на терминале идентифицировать товар (находить по сканированному штрихкоду, выбирать из списка, находить по наименованию). Если просто сканировать штрихкоды без идентификации товара, то у оператора не будет возможности прямо на месте узнать: известен ли этот штрихкод учетной системе? верно ли заведена карточка товара? верная ли цена?

Сидя у компьютера будет очень трудно понять, к чему относятся ошибки загрузки результатов в учетную систему. Ошибки надо выдавать прямо на ТСД еще во время сканирования. Иначе на большом экране ПК будет список ошибок и будет написано “*неизвестный штрихкод 1234567890*”. А как понять, что это был за товар в торговом зале? Мы целые пол часа ходили сканировали собирали штрихкоды.

Кроме того, в каждой строке выгружаемых на терминал документов-заданий должен быть указан идентификатор товара и упаковки, которые обязаны соответствовать идентификаторам позиций выгруженного в базу Mobile SMARTS справочника номенклатуры.

В каждой учетной системе справочник номенклатуры имеет свою структуру. В общем случае из учетной системы может выгружаться не обязательно “номенклатура”, это могут быть “основные средства”, “инвентарные позиции”, “детали” и т.п. Любой большой список позиций, для которых требуется идентификация и быстрый поиск с занесением в строку документа. Перед написанием кода для выгрузки номенклатуры из учетной системы в Mobile SMARTS необходимо решить, как отобразится структура данных из учетной системы на объекты Mobile SMARTS.

Для «1С:Предприятия» выгрузку номенклатуры лучше всего проводить через методы TerminalConnector, которые не только принимают таблицы значений, но и поддерживают самые последние стандарты 1С для драйверов торгового оборудования.

Для остальных систем необходимо работать с ComConnector и объектами номенклатуры, которые в нем представлены.

Объектом, представляющим товар в Mobile SMARTS является Product [Товар], каждый товар содержит

коллекцию упаковок Packing [Упаковка] (у товара должна быть хотя бы одна упаковка). У товара должна быть установлена базовая упаковка при помощи свойства BasePackingId [ИдБазовойУпаковки]. Базовая упаковка должна содержатся в коллекции упаковок Packings [Упаковки]. Базовая упаковка обычно имеет коэффициент (количество единиц UnitsQuantity [КоличествоБазовыхЕдиниц]) равный 1. Для остальных упаковок (например, коробки, паллеты) UnitsQuantity [КоличествоБазовыхЕдиниц] определяет, сколько базовых упаковок содержится в данной (например, в коробке 10 шт, в паллете 500 шт).

Если товар весовой и упаковок в действительности нет, при выгрузке в Mobile SMARTS должна быть создана одна упаковка (например, Килограммы (кг)) и она же назначена базовой.

Если различать упаковки товара нет необходимости (в учетной системе есть только сущность “Товар”), также нужно создать одну упаковку (например, “шт”) и назначить ее базовой.

В учетной системе могут быть различные сущности, связанные с товаром отношением “один ко многим” и идентифицируемые по штрихкоду, но не являющиеся упаковками. Например, серии и характеристики товара. Эти сущности в Mobile SMARTS также могут быть отражены как упаковки товара. (Примечание: можно использовать и дополнительные таблицы, которые определяются разработчиком конфигурации Mobile SMARTS, и связь с номенклатурой через ключ связи (Идентификатор товара), но при использовании упаковок уже будут работать все готовые механизмы по выбору номенклатуры, быстрому поиску по штрихкоду, записи в документ).

Объекты Product [Товар] и Packing [Упаковка] содержат как основные свойства, такие как Id [Ид], Name [Имя], Barcode [Штрихкод], Marking [Артикул], которые есть всегда в объектах данного типа, так и дополнительные, определяемые разработчиком конкретной конфигурации Mobile SMARTS. См. ниже Дополнительные поля.

Основные свойства и методы:
Product [Товар]

Свойства		
Наименование	**Тип**	**Описание**
Id [Ид]	Строка	Уникальный идентификатор товара.
Name [Имя]	Строка	Наименование товара.
Barcode [Штрихкод]	Строка	Основной штрихкод товара. Определяет только товар, без конкретной упаковки. Поиск при сканировании на терминале в первую очередь осуществляется по этому штрихкоду. Если товар найден и имеет несколько упаковок - то пользователю будет предложен выбор из списка, в какой упаковке сканирован штрихкод. Штрихкодов может быть задано несколько, с помощью разделителя '|'.
Marking [Артикул]	Строка	Общий артикул товара (без указания конкретной упаковки). Если необходимо, чтобы один и тот же товар в разных упаковках имел разные артикулы используйте такое же свойство у упаковки: Marking [Артикул].

BasePackingId [ИдБазовойУпаковки]	Строка	Идентификатор основного типа упаковки. Упаковка с данным идентификатором должна быть в коллекции упаковок товара Packings [Упаковки]. Подробнее см. Packing [Упаковка].
Packings [Упаковки]	PackingCollection [КоллекцияУпаковок]	Коллекция типов упаковок для товара. Подробнее см. Packing [Упаковка].
Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**

GetField [ПолучитьПоле]	Параметр: string (имя поля)
	Возвращаемое значение: object (значение поля)
		Возвращает значение дополнительного поля. Если поля с переданным именем в объекте нет, вернется null.
		Пример:
		`var value = product.GetField("ИмяПоля");`
		
		`значПоля = товар.ПолучитьПоле("ИмяПоля");`

SetField [УстановитьПоле]	Параметры: string (имя поля), object (значение)
	Возвращаемое значение: нет
		Устанавливает значение дополнительного поля.
		Пример:
		`product.SetField("ИмяПоля", value);`
		
		`товар.УстановитьПоле("ИмяПоля", значение);`

Packing [Упаковка]

Свойства		
Наименование	**Тип**	**Описание**
Id [Ид]	Строка	Уникальный идентификатор типа упаковки. Уникальность в пределах упаковок одного товара.
Barcode [Штрихкод]	Строка	Штрихкод упаковки. Определяет товар в конкретной упаковке. При сканировании после поиска по основному штрихкоду товара, начинается поиск по этому штрихкоду. Если штрихкод найден, то выбирает товар сразу в конкретной упаковке. Кроме обычного задания в виде строки из цифр и букв, может задаваться в виде шаблона из комбинации символов и блоков

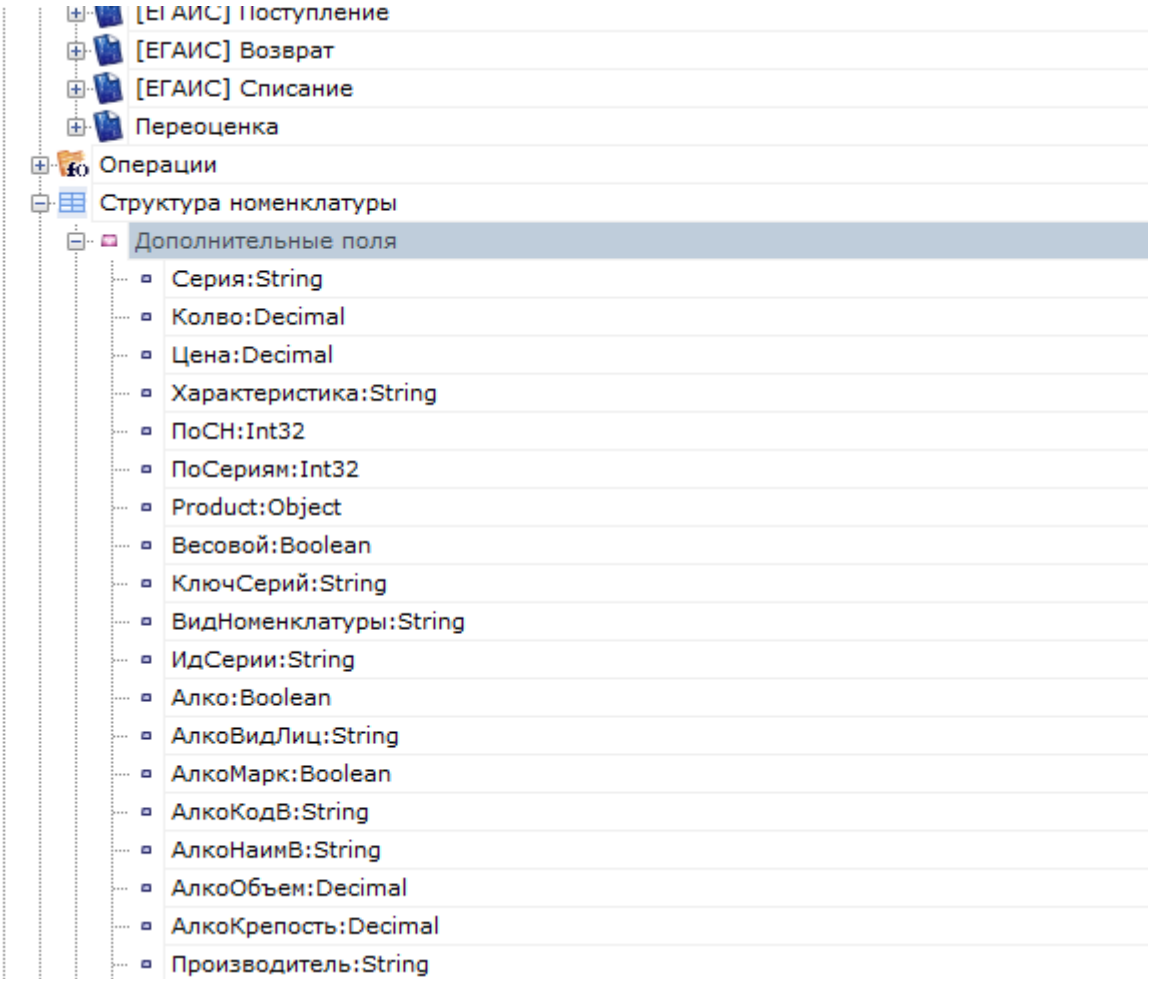
		{Template}. Такие блоки служат для вставки в штрихкод количества, срока годности и так далее. Подробнее про применение шаблонов штрихкодов см. тут. Штрихкодов и шаблонов может быть задано несколько, с помощью разделителя '|'.
Marking [Артикул]	Строка	Артикул товара в данной упаковке. Заполняется, если для упаковок товара используются артикулы, отличные от артикула товара.
Name [Имя]	Строка	Наименование упаковки.
UnitsQuantity [КоличествоБазовыхЕдиниц]	Число	Количество единиц товара в упаковке.
Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**

GetField [ПолучитьПоле]	Параметр: string (имя поля)
	Возвращаемое значение: object (значение поля)
		Возвращает значение дополнительного поля. Если поля с переданным именем в объекте нет, вернется null.
		Пример:
		`var value = pack.GetField("ИмяПоля");`
		
		`значПоля = упаковка.ПолучитьПоле("ИмяПоля")`

SetField [УстановитьПоле]	Параметры: string (имя поля), object (значение)
	Возвращаемое значение: нет
		Устанавливает значение дополнительного поля.
		Пример:
		`pack.SetField("ИмяПоля", value);`
		
		`упаковка.УстановитьПоле("ИмяПоля", значение);`

Дополнительные поля

При разработке конфигурации Mobile SMARTS могут быть определены произвольные дополнительные поля номенклатуры. Например, поле “Производитель”, в которое можно записывать при выгрузке номенклатуры наименование производителя товара или поле “Остаток”, в которое будет выгружаться количество данного товара на складе. Дополнительные поля задаются в Панеле управления Mobile SMARTS (см. Панель управления) в разделе Структура номенклатуры:



Заданные поля могут использоваться как для объектов Product [Товар], так и для Packing [Упаковка].
Для упаковки, например, могут выгружаться поля “Серия” и “Характеристика”, в которых будут наименование серии и характеристики товара (в случае, если упаковки определяют серии и характеристики товара).
Для выгрузки используются следующие функции объекта `Cleverence.Warehouse.StorageConnector`:

StorageConnector [Соединение]

Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**
BeginUploadProducts [НачатьПорционнуюВыгрузкуНоменклатуры]	Параметры: bool *anyway,* *bool overwriteExisting,* *bool generateFullTextSearch* *anyway* Начать новую выгрузку даже если какая-то другая выгрузка была открыта. *overwriteExisting* Флаг, задающий надо ли полностью заменить номенклатуру в базе Mobile SMARTS или слить с существующей. *generateFullTextSearch* Флаг, задающий нужно ли	Функция открывает выгрузку номенклатуры в базу Mobile SMARTS.

	генерировать индекс для полнотекстового поиска номенклатуры по наименованию. Возвращаемое значение: нет	
UploadProducts [ВыгрузитьПорциюНоменклатуры]	Параметры: ProductCollection *products* *products* Коллекция позиций номенклатуры, которые должны быть добавлены в текущую выгрузку. Возвращаемое значение: нет	Выгружает номенклатуру в базу Mobile SMARTS в порционном режиме. Учитывайте, что справочник номенклатуры реально изменится ТОЛЬКО после вызова EndUploadProducts [ЗавершитьПорционнуюВыгрузкуНоменклатуры].
EndUploadProducts [ЗавершитьПорционнуюВыгрузкуНоменклатуры]	Параметры: нет Возвращаемое значение: нет	Завершение порционной выгрузки номенклатуры. После этого вызова вся накопленная с помощью вызовов UploadProducts[ВыгрузитьПорциюНоменклатуры] номенклатура будет сохранена в базе Mobile SMARTS.
ResetUploadProducts [СброситьПорционнуюВыгрузкуНоменклатуры]	Параметры: нет Возвращаемое значение: нет	Сброс режима порционной выгрузки. Имеет смысл использовать для серверной базы Mobile SMARTS. Все выгруженные, но не подтвержденные завершающим вызовом EndUploadProducts [ЗавершитьПорционнуюВыгрузкуНоменклатуры] товары не сохранятся в базе Mobile SMARTS на сервере. Если база не серверная, функция ничего не делает. Функцию нужно использовать, если выгрузка началась с BeginUploadProducts [НачатьПорционнуюВыгрузкуНоменклатуры] и далее в процессе выгрузки возникли ошибки. Функция сообщает серверу, что выгрузку нужно прервать (чтобы можно было впоследствии начать следующую выгрузку). Для несерверной базы прерывание выгрузки не требуется, данные в любом случае не сохранятся без вызова EndUploadProducts [ЗавершитьПорционнуюВыгрузкуНоменклатуры].

Выгрузка называется “порционной”, т.к. нет необходимости формировать весь список выгружаемых

товаров сразу и передавать его в выгрузку целиком. Можно набрать некоторое количество товаров в коллекцию ProductCollection *products,* выполнить вызов UploadProducts [ВыгрузитьПорциюНоменклатуры], набрать еще и т.д. до обхода всех товаров, которые нужно выгрузить.

Общий алгоритм выгрузки номенклатуры. В общем виде алгоритм выгрузки выгладит следующим образом:

1. Создаем объект Cleverence.Warehouse.StorageConnector и устанавливаем подключение к нужной базе Mobile SMARTS при помощи вызова SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС];
2. Начинаем выгрузку с помощью вызова BeginUploadProducts [НачатьПорционнуюВыгрузкуНоменклатуры];
3. Создаем объект ProductCollection [КоллекцияТоваров];
4. Получаем в учетной системе выборку записей, содержащую нужные поля для заполнения объектов Product [Товар] и Packing [Упаковка];
5. Обходим в цикле выборку записей, создаем объекты Product [Товар] и Packing [Упаковка], упаковки добавляем в коллекцию упаковок Packings [Упаковки] объекта Product, обязательно указываем базовую упаковку (BasePackingId [ИдБазовойУпаковки]), добавляем объекты Product [Товар] в коллекцию товаров productCollection;
6. Если в коллекции товаров набралось достаточно много записей (например, 1000), вызываем выгрузку порции товаров UploadProducts [ВыгрузитьПорциюНоменклатуры]. Создаем новый объект ProductCollection [КоллекцияТоваров], в который на следующих итерациях цикла будем накапливать товары;
7. После завершения цикла, если в коллекции товаров есть записи, вызываем UploadProducts [ВыгрузитьПорциюНоменклатуры];
8. Завершаем выгрузку, вызвав EndUploadProducts [ЗавершитьПорционнуюВыгрузкуНоменклатуры].

Пример
Допустим в результате некоторого отбора из различных таблиц в учетной системе мы получили выборку со следующими полями:

itemId (Идентификатор позиции номенклатуры)
name (наименование позиции номенклатуры)
marking (артикул)
unit (наименование единицы измерения, “шт”, “кг” и т.д.)
factor (коэффициент пересчета в базовые для данной единицы)
barcode (штрихкод)
descr (наименование характеристики номенклатуры)
qty (количество на складе)
price (цена)

В результате выборки получаем:

itemId	name	marking	unit	factor	barcode	descrl	qty	price
0001213	Джемпер	Д-0920	шт	1				
0001213	Джемпер	Д-0920	шт	1	2500001780331	26/92, красный	5	359,00
0001213	Джемпер	Д-0920	шт	1	2500001780263	24/80, красный	7	359,00
0001213	Джемпер	Д-0920	шт	1	2500001780347	26/92, голубой	1	400,00
00000049	Бренди		шт	1	2200007761321		10	1200,00

00000049	Бренди		упак	6	2200007761321			7000,00
00000097	Крупа	Арт-8900	кг	1	10055		20	51,00
00000070	Печенье	Арт-4444	упак	1	2000020577788		8	62,00
00000070	Печенье	Арт-4444	упак	1	4607038271677		8	62,00

В выборке для одного и того же товара может быть несколько строк, т.к. может быть несколько характеристик, упаковок или штрихкодов одного товара.

Выгрузка:

C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта соединения

// Выполняем подключение к базе Mobile SMARTS
connector.SelectCurrentApp("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
try
{
  connector.BeginUploadProducts(true, true, true); //Начинаем выгрузку, в параметрах указываем, что
  //справочник перезаписываем и создаем индекс для поиска по наименованию
  var products = new Cleverence.Warehouse.ProductCollection(); //Создаем коллекцию для накопления
  //выгружаемых товаров

  SqlCommand command = new SqlCommand();
  command.CommandText = query; //Запрос к БД для чтения данных

   using (SqlConnection connection = new SqlConnection(connectionString)) //Устанавливаем соединение с //БД,
  из которой будем читать данные
   {
     connection.Open();
     command.Connection = connection;
     SqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) //Читаем данные
      {
        string itemId = reader.GetString(0);
        string name = reader.GetString(1);
        string marking = reader.GetString(2);
        string unit = reader.GetString(3);
        decimal factor = reader.GetDecimal(4);
        string barcode = reader.GetString(5);
        string descr = reader.GetString(6);
        decimal qty = reader.GetDecimal(7);
        decimal price = reader.GetDecimal(8);
        //Получили все поля одной записи

        Cleverence.Warehouse.Product uploadProduct = products.FindById(itemId); //Ищем товар по id //среди
        набранных к выгрузке товаров
        if(uploadProduct == null)
        {
          //Если товара нет, нужно будет создать новый объект и добавить его в коллекцию.
          //Перед этим проверяем, не набрали ли мы уже достаточное количество товаров.
          if(products.Count >= 1000)
          {
             //Если набрали достаточно, выгружаем порцию товаров и создаем новый объект коллекции, чтобы
             //набирать товары дальше в пустую коллекцию.
             connector.UploadProducts(products);
             products = new Cleverence.Warehouse.ProductCollection();
          }

          uploadProduct = new Cleverence.Warehouse.Product();
```

```
    uploadProduct.Id = itemId;
    uploadProduct.Name = name;
    uploadProduct.Marking = marking;

    products.Add(uploadProduct);
  }
}

Cleverence.Warehouse.Packing uploadPacking = null;
//Проверяем, если упаковка с той же единицей, характеристикой и ценой есть в упаковках данного
//товара, добавлять новую упаковку не нужно, нужно будет только добавить в эту же упаковку //штрихкод.
foreach(Cleverence.Warehouse.Packing pack in uploadProduct.Packings)
{
   if(pack.Name == unit && pack.GetField("Характеристика") == descr
    && pack.GetField("Цена") == price)
   {
      uploadPacking = pack;
      break;
   }
}

if(uploadPacking == null)
{
  //Такой упаковки нет, создаем новую
  uploadPacking = new Cleverence.Warehouse.Packing();
  //Для упаковки нужно использовать уникальный идентификатор в пределах упаковок данного товара
  string packId = unit;
  if(uploadProduct.Packings.FindById(unit) != null)
  {
     //Оказалось, что упаковка с Ид. равным единице измерения уже есть (могли быть разные
     //характеристики товара с одной единицей).
     //Генерируем новый Ид. вида [unit]_[номер] (например, "шт_1", "шт_2" и т.д.)
     int cnt = 0;
     foreach (Packing p in uploadProduct.Packings)
       if (p.Id.StartsWith(unit + "_"))
          cnt++;

     packId = String.Format("{0}_{1}", unit, cnt + 1);
  }

  uploadPacking.Id = packId;
  uploadPacking.Name = unit; //Имя равно единице измерения
  uploadPacking.UnitsQuantity = factor; //Коэффициент пересчета в базовые единицы
  uploadPacking.Barcode = barcode; //Штрихкод
  uploadPacking.SetField("Характеристика", descr); //Устанавливаем доп. поля
  uploadPacking.SetField("КоличествоНаСкладе", qty);
  uploadPacking.SetField("Цена", price);

  if(factor == 1.0 && uploadProduct.BasePackingId == null)
  {
     //Если коэффициент равен 1 и базовую упаковку еще не установили, считаем данную упаковку //базовой.
     uploadProduct.BasePackingId = packId;
  }

  uploadProduct.Packings.Add(uploadPacking); //Добавляем упаковку в коллекцию упаковок товара
}
else
{
  //Если такая упаковка уже есть, добавляем к ней новый штрихкод (если он не пустой)
  if(!string.IsNullOrEmpty(barcode))
    uploadPacking.Barcode = uploadPacking.Barcode + "|" + barcode;
```

```
      }

   }

   if(products.Count > 0)
   {
     //Если остались невыгруженные товары, их нужно выгрузить
     connector.UploadProducts(products);
   }

   connector.EndUploadProducts(); //Завершаем выгрузку товаров.
}
catch
{
  // При выгрузке возникла ошибка, обрабатываем исключение и сбрасываем начавшуюся выгрузку.
  // В реальном приложении нужна отдельная обработка исключений, связанных с чтением данных из базы SQL // и
  с выгрузкой в Mobile SMARTS.
  connector.ResetUploadProducts();
}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
//соединения
Попытка
    connector.УстановитьПодключениеСБазойСМАРТС("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
  //Выполняем подключение

  Запрос = Новый Запрос(ТекстЗапроса);
  ТаблицаТоваров = Запрос.Выполнить().Выгрузить(); //Выполняем запрос

  connector.BeginUploadProducts(Истина, Истина, Истина); //Начинаем выгрузку, в параметрах указываем, //что
  справочник перезаписываем и создаем индекс для поиска по наименованию
  products = новый COMОбъект("Cleverence.Warehouse.ProductCollection"); //Создаем коллекцию для //накопления
  выгружаемых товаров

  Для Каждого СтрокаТаблицы из ТаблицаТоваров Цикл
     ИдНоменклатуры = XMLСтрока(СтрокаТаблицы.НоменклатураСсылка); //В качестве идентификатора товара //будем
     выгружать Guid позиции номенклатуры
     Наименование = СтрокаТаблицы.НоменклатураНаименование;
     Артикул = СтрокаТаблицы.Артикул;
     ЕдиницаИзмерения = СтрокаТаблицы.УпаковкаНаименование;
     Коэффициент = СтрокаТаблицы.УпаковкаКоэффициент;
     Штрихкод = СтрокаТаблицы.Штрихкод;
     Характеристика = СтрокаТаблицы.ХарактеристикаНаименование;
     Остаток = СтрокаТаблицы.ОстатокНаСкладе;
     Цена = СтрокаТаблицы.Цена;

     uploadProduct = products.НайтиПоИд(ИдНоменклатуры); //Ищем товар по id среди набранных к выгрузке
     //товаров
     Если uploadProduct = Неопределено Тогда
      //Если товара нет, нужно будет создать новый объект и добавить его в коллекцию.
      //Перед этим проверяем, не набрали ли мы уже достаточное количество товаров.
        Если products.Количество >= 1000 Тогда
           //Если набрали достаточно, выгружаем порцию товаров и создаем новый объект коллекции, чтобы
           //набирать товары дальше в пустую коллекцию.
           connector.ВыгрузитьПорциюНоменклатуры(products);
           products = новый COMОбъект("Cleverence.Warehouse.ProductCollection");
        КонецЕсли;

        uploadProduct = новый COMОбъект("Cleverence.Warehouse.Product");
```

```
        uploadProduct.Ид = ИдНоменклатуры;
        uploadProduct.Имя = Наименование;
        uploadProduct.Артикул = Артикул;

        products.Добавить(uploadProduct);
    КонецЕсли;

    uploadPacking = Неопределено;
    //Проверяем, если упаковка с той же единицей, характеристикой и ценой есть в упаковках данного
    //товара, добавлять новую упаковку не нужно, нужно будет только добавить в эту же упаковку //штрихкод.
    Для Инд=0 По uploadProduct.Упаковки.Количество-1 Цикл
        pack = uploadProduct.Упаковки.Элемент(Инд);
        Если pack.Name = unit И pack.ПолучитьПоле("Характеристика") == Характеристика
         И pack.ПолучитьПоле("Цена") == Цена Тогда
            uploadPacking = pack;
            Прервать;
        КонецЕсли;
    КонецЦикла;

    Если uploadPacking = Неопределено Тогда
      //Такой упаковки нет, создаем новую
      uploadPacking = новый COMОбъект("Cleverence.Warehouse.Packing");
      //Для упаковки нужно использовать уникальный идентификатор в пределах упаковок данного товара
      packId = ЕдиницаИзмерения;
      Если uploadProduct.Упаковки.НайтиПоИд(ЕдиницаИзмерения) <> Неопределено Тогда
        //Оказалось, что упаковка с Ид. равным единице измерения уже есть (могли быть разные
        //характеристики товара с одной единицей).
        //Генерируем новый Ид. вида [ЕдиницаИзмерения]_[номер] (например, "шт_1", "шт_2" и т.д.)
        Сч = 0;
        Для Инд=0 По uploadProduct.Упаковки.Количество-1 Цикл
            p = uploadProduct.Упаковки.Элемент(Инд);
          Если (p.Ид.СтрНачинаетсяС(ЕдиницаИзмерения + "_")) Тогда
              Сч++;
          КонецЕсли;
        КонецЦикла;
        packId = ЕдиницаИзмерения + "_" + (Сч + 1);
      КонецЕсли;

      uploadPacking.Ид = packId;
      uploadPacking.Имя = ЕдиницаИзмерения; //Имя равно единице измерения
      uploadPacking.КоличествоБазовыхЕдиниц = Коэффициент; //Коэффициент пересчета в базовые единицы
      uploadPacking.Штрихкод = Штрихкод;
      uploadPacking.УстановитьПоле("Характеристика", Характеристика); //Устанавливаем доп. поля
      uploadPacking.УстановитьПоле("КоличествоНаСкладе", Остаток);
      uploadPacking.УстановитьПоле("Цена", Цена);

      Если Коэффициент = 1.0 И uploadProduct.ИдБазовойУпаковки = Неопределено Тогда
        //Если коэффициент равен 1 и базовую упаковку еще не установили, считаем данную упаковку //базовой.
        uploadProduct.ИдБазовойУпаковки = packId;
      КонецЕсли;

      uploadProduct.Упаковки.Добавить(uploadPacking); //Добавляем упаковку в коллекцию упаковок //товара
    Иначе
      //Если такая упаковка уже есть, добавляем к ней новый штрихкод (если он не пустой)
      Если ЗначениеЗаполнено(Штрихкод) Тогда
        uploadPacking.Штрихкод = uploadPacking.Штрихкод + "|" + Штрихкод;
      КонецЕсли;
    КонецЕсли;
КонецЦикла;

  Если products.Количество > 0 Тогда
```

```
    //Если остались невыгруженные товары, их нужно выгрузить
    connector.ВыгрузитьПорциюНоменклатуры(products);
  КонецЕсли;

  connector.EndUploadProducts (); //Завершаем выгрузку товаров.

Исключение
// При выгрузке возникли ошибки, обрабатываем исключение
  connector.СброситьПорционнуюВыгрузкуНоменклатуры();
КонецПопытки;
```

Обмен документами

При выполнении любых операций на терминале (Приемка, Отгрузка, Инвентаризация и др.) пользователь работает с некоторым документом Mobile SMARTS. Документ может быть выгружен из учетной системы и содержать список товаров, которые нужно отсканировать (или предполагается, что они будут отсканированы). Например, должна произойти приемка товара на склад и из учетной системы выгружается накладная со списком товаров, при этом по факту приемки могут быть расхождения (какой-то товар не привезли или привезли не в том количестве, а может придти товар, которого нет по накладной). Фактически отсканированный товар фиксируется в документе Mobile SMARTS. После завершения работы с документом на терминале, документ загружается в учетную систему и происходит регистрация принятого товара. Кроме выгрузки из учетной системы, документ может быть создан на терминале пользователем (если это позволяют настройки данного типа документа, заданные в конфигурации Mobile SMARTS). В таком документе будут только фактически отсканированные товары. Например, приемка на склад может производится по факту без выгрузки какого-либо документа из учетной системы. В этом случае при загрузке завершенного документа с терминала, как правило, создается новый документ учетной системы.

Таким образом, документ Mobile SMARTS служит заданием для выполнения на терминале (если он выгружается из учетной системы) и является результатом работы пользователя на терминале, который требуется отразить в учетной системе.
Алгоритм работы с документом каждого типа (какие данные должен вводить пользователь и как они будут обрабатываться), задается в Панели управления (см. Панель управления, Тип документа).

Документ Mobile SMARTS содержит два списка товаров (Плановая часть - товары, которые должны быть или предполагается отсканировать, Фактическая часть - фактически отсканированные товары), а также, если необходимо, другие данные (поля шапки, дополнительные табличные части).
Объектом документа является Document [Документ]. С помощью этого объекта происходит обмен данными между учетной системой и Mobile SMARTS (выгрузка документов-заданий и загрузка результатов работы с терминалов).
Document [Документ] содержит следующие основные поля и методы:

Свойства		
Наименование	**Тип**	**Описание**
Свойства шапки документа, назначаемые при выгрузке из учетной системы:		
Id [Ид]	Строка	Уникальный идентификатор документа. При выгрузке из учетной системы должен идентифицировать исходный выгружаемый документ. При создании документа на терминале назначается автоматически.
Name [Имя]	Строка	Название документа. Произвольное наименование, которое отображается

		на экране терминала. Назначается при выгрузке из учетной системы (например, “Приемка №327 от 12.07.16”), при создании документа на терминале присваивается автоматически.
DocumentTypeName [ИмяТипаДокумента]	Строка	Должно соответствовать имени одного из типов документов, имеющихся в конфигурации Mobile SMARTS.
WarehouseId [ИдСклада]	Строка	Идентификатор склада Mobile SMARTS, к которому привязан документ. При выгрузке из учетной системы должен быть равен Ид. одного из складов, имеющихся в конфигурации Mobile SMARTS. В типовых конфигурациях по умолчанию существует единственный склад “Общий” с Ид. равным “1”.
Appointment [Назначение]	Строка	Назначение документа - код пользователя или имя группы пользователей, которым данный документ назначается на исполнение. Если значение пустое, то документ попадает к первому свободному пользователю, которому разрешен тип документа. В типовых конфигурациях Mobile SMARTS по умолчанию заведен единственный пользователь с Ид. “оператор”.
AutoAppointed [ВыдаватьАвтоматически]	Булево	Флаг автоматической выдачи. Если стоит Истина - документ выдается на ТСД, автоматически, не дожидаясь явного выбора его пользователем из списка или по ШК. Имеет смысл только для серверной базы Mobile SMARTS. По умолчанию Истина. Выдача автоматически не означает, что документ будет сразу же открыт без участия пользователя. Сервер назначает документ для исполнения, он отправляется на терминал, пользователь получает звуковое уведомление, дальше документ нужно будет открыть, выбрав его из списка или по ШК.
Barcode [Штрихкод]	Строка	Штрихкод документа. Документ на терминале может быть открыт путем сканирования штрихкода с бумажной копии документа (если это разрешено настройками типа документа в конфигурации Mobile SMARTS).
ServerHosted [ИсполняемыйНаСервере]	Булево	Признак того, что документ должен выполняться "на сервере". Такой документ могут одновременно открыть на редактирвоание несколько пользователей. Все изменения в документе будут происходить

		одновременно для всех работающих с ним пользователей. Работа в таком режиме требует наличия постоянной связи с сервером. По умолчанию Ложь.
CreateDate [ДатаСоздания]	ДатаВремя	Дата создания документа. Если не назначать, проставится текущая дата.
Priority [Приоритет]	Целое	Приоритет документа. Более приоритетные документы раньше отдаются на терминал для обработки. Имеет смысл только для серверной базы Mobile SMARTS. Назначать не обязательно.
Description [Описание]	Строка	Описание документа, назначать не обязательно.
Свойства шапки документа, которые могут использоваться при загрузке в учетную систему:		
Finished [Завершен]	Булево	Признак того, что обработка документа пользователем была завершена, и его можно забирать назад в учетную систему.
Modified [Изменен]	Булево	Признак того, что документ был изменен.
InProcess [ВОбработке]	Булево	Признак того, что документ захвачен пользователем на обработку.
Completed [ВсеСтрокиПланаВыполнены]	Булево	Свойство, позволяющее проверить все ли строки задания выполнены (фактическое количество товара равно плановому). Проверка происходит по плановым строкам DeclaredItems.
Overloaded [ЕстьПереполнение]	Булево	Признак, есть ли превышение количества товара в строках документа.
Underloaded [ЕстьНедобор]	Булево	Признак, есть ли недобор количества товара в строках документа.
CreatedOnPDA [СозданНаТСД]	Булево	Признак того, что документ был создан на ТСД. Истина - создан на ТСД, Ложь - выгружен из учетной системы.
UserId [ИдПользователя]	Строка	Ид. пользователя ТСД, который выполнил документ.
UserName [ИмяПользователя]	Строка	Имя пользователя ТСД, который выполнил документ.
DeviceId [ИдУстройства]	Строка	Ид. устройства (уникальный код терминала), на котором был завершен документ.
DeviceIP [ИпУстройства]	Строка	IP-адрес устройства, на котором был завершен документ.
DeviceName [ИмяУстройства]	Строка	Имя устройства, на котором был

		завершен документ.
Коллекции строк документа:		
DeclaredItems [СтрокиПлан]	DocumentItemCollection [КоллекцияСтрокДокумента]	Коллекция строк документа DocumentItem [СтрокаДокумента], содержащая данные о товарах, которые предполагается отсканировать по плану. Заполняется при выгрузке из учетной системы. При этом на терминале при работе с документом также может происходить запись в плановые строки.
CurrentItems [СтрокиФакт]	DocumentItemCollection [КоллекцияСтрокДокумента]	Коллекция строк документа DocumentItem [СтрокаДокумента], содержащая данные о фактически отсканированных товарах. Заполняется при работе на терминале.
Tables [Таблицы]	DocumentTableCollection [КоллекцияТаблиц]	Коллекция дополнительных таблиц документа. В конфигурации Mobile SMARTS для типа документа могут быть определены дополнительные табличные части с произвольным набором полей. Дополнительные табличные части могут быть заполнены при выгрузке документа из учетной системы, также они могут заполняться при работе на терминале.
Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**

GetField [ПолучитьПоле]	Параметр: string (имя поля)
	Возвращаемое значение: object (значение поля)
		Возвращает значение дополнительного поля шапки документа. Если поля с переданным именем в объекте нет, вернется null.
		Пример:
		`var value = document.GetField("ИмяПоля");`
		
		`значПоля = документ.ПолучитьПоле("ИмяПоля")`

SetField [УстановитьПоле]	Параметры: string (имя поля), object (значение)
	Возвращаемое значение: нет
		Устанавливает значение дополнительного поля шапки документа.
		Пример:
		`document.SetField("ИмяПоля", value);`
		
		`документ.УстановитьПоле("ИмяПоля", значение);`

Объектом строки документа является DocumentItem [СтрокаДокумента]. Данные объекты содержатся в коллекции плановых (DeclaredItems [СтрокиПлан]) и фактических (CurrentItems [СтрокиФакт]) строк документа.
DocumentItem [СтрокаДокумента] содержит следующие основные поля и методы:

Свойства		
Наименование	**Тип**	**Описание**
ProductId [ИдТовара]	Строка	Идентификатор товара. Товар с таким идентификатором должен быть в выгруженном в справочнике номенклатуры.
PackingId [ИдУпаковки]	Строка	Идентификатор упаковки товара. Упаковка с таким ид. должна быть в коллекции упаковок товара, заданного в строке документа.
DeclaredQuantity [КоличествоПлан]	Число	Плановое количество товара. Значение заполняется при выгрузке документа из учетной системы.
CurrentQuantity [КоличествоФакт]	Число	Фактическое количество товара. Заполняется при работе на терминале и отражает, сколько фактически было отсканировано данного товара.
SSCC [КодЕдиницыХранения]	Строка	Serial shipping container code - уникальный номер единицы хранения. Поле может использоваться для занесения номера контейнера, паллеты и т.д. Может заполняться из штрихкода EAN-128 (см. Интерпретация кода EAN-128 в Mobile SMARTS).
FirstStorageBarcode [ШтрихкодПервогоМеста]	Строка	Штрихкод ячейки, паллеты и т.п., в которых находится или из которых перемещается товар.
SecondStorageBarcode [ШтрихкодВтороГоМеста]	Строка	Штрихкод ячейки, паллеты и т.п. в которые перемещается товар из первого места хранения.
ExpiredDate [СрокГодности]	ДатаВремя	Срок годности товара. Может заполняться из штрихкода EAN-128 (см. Интерпретация кода EAN-128 в Mobile SMARTS).
RegistrationDate [ДатаРегистрации]	ДатаВремя	Дата регистрации товара. Может заполняться на терминале.
BindedLine [СвязаннаяСтрока]	DocumentItem [СтрокаДокумента]	Связанная строка документа. Заполняется на терминале и задает связь строки из фактической части со строкой из плановой части документа.
Overload [Переполнение]	Число	Возвращает превышение фактического количества товара над плановым (разницу между фактическим и плановым количеством).

Underload [Недобор]	Число	Возвращает разницу между плановым и фактическим количеством.
UnderloadedOrOverloaded [ЕстьНедоборИлиПереполнение]	Булево	Позволяет проверить совпадают ли заявленное и фактическое количество в строке. Истина - количества разные, Ложь - количества одинаковые.
Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**

GetField [ПолучитьПоле]	Параметр: string (имя поля)
	Возвращаемое значение: object (значение поля)
		Возвращает значение дополнительного поля шапки документа. Если поля с переданным именем в объекте нет, вернется null.
		Пример:
		`var value = documentItem.GetField("ИмяПоля");`
		
		`значПоля = строкаДокумента.ПолучитьПоле("ИмяПоля")`

SetField [УстановитьПоле]	Параметры: string (имя поля), object (значение)
	Возвращаемое значение: нет
		Устанавливает значение дополнительного поля шапки документа.
		Пример:
		`documentItem.SetField("ИмяПоля", value);`
		
		`строкаДокумента.УстановитьПоле("ИмяПоля", значение);`

Для обмена документами используются следующие функции объекта **`StorageConnector:`**

StorageConnector [Соединение]

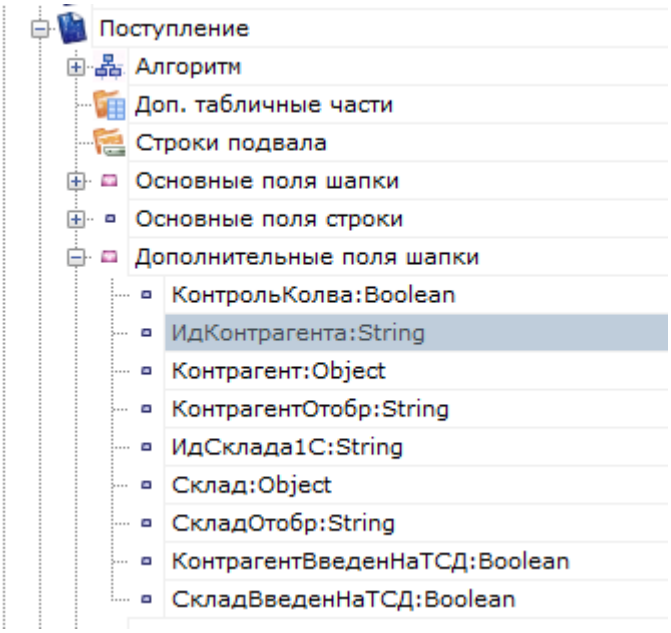
Методы		
Наименование	**Параметры и возвращаемое значение**	**Описание**
Выгрузка документов:		
SetDocument [ВыгрузитьДокумент]	Параметры: Document *document* *document* Выгружаемый документ. Возвращаемое значение: нет	Выгружает документ в базу Mobile SMARTS.
SetDocuments [ВыгрузитьДокументы]	Параметры: DocumentCollection *documents*	Выгружает сразу несколько документов в базу Mobile SMARTS.

	documents Коллекция выгружаемых документов. Возвращаемое значение: нет	
Получение документов:		
GetDocuments [ПолучитьДокументы]	Параметры: string docType, bool checkForFinish docType Наименование типа документов, которые следует получить. Если передать пустую строку, возвращает все документы, независимо от типа. checkForFinish Если Истина, вернутся только завершенные на терминале документы. Если Ложь, все документы. Возвращаемое значение: Коллекция документов, отобранных в соответствии с заданными условиями.	Получает документы из базы Mobile SMARTS в соответствии с заданными параметрами. Если документов, удовлетворяющих условиям нет, вернется пустая коллекция.
GetDocument [ПолучитьДокумент]	Параметры: *string documentId* *documentId* Идентификатор документа. Возвращаемое значение: объект документа Document или null.	Получает документ из базы Mobile SMARTS по идентификатору.
GetDocumentsByIds [ПолучитьДокументыПоИдентификаторам]	Параметры: *StringCollection idList* *idList* Коллекция идентификаторов документов. Возвращаемое значение: коллекция документов DocumentCollection.	Получает из базы Mobile SMARTS документы с заданными идентификаторами.
GetDocument [ПолучитьДокумент]	Параметры: *string idocumentId* *idocumentId* Идентификатор документа. Возвращаемое значение: объект документа Document или null, если документ не найден.	Получает документ из базы Mobile SMARTS по идентификатору.

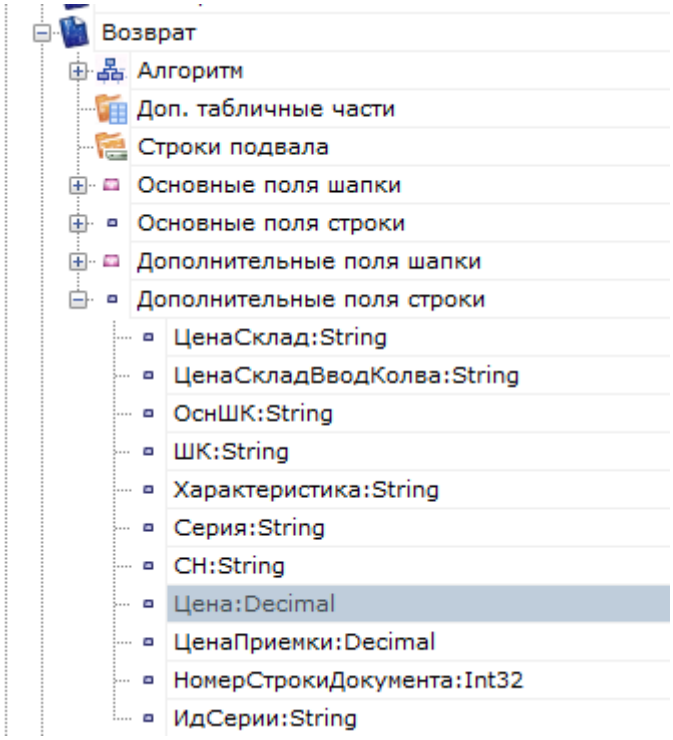
Удаление документов:		
RemoveDocument [УдалитьДокумент]	Параметры: *string documentId* *documentId* Идентификатор документа. Возвращаемое значение: нет.	Удаляет документ из базы Mobile SMARTS по идентификатору. Если документ не найден, функция ничего не делает.
RemoveDocuments [УдалитьДокументы]	Параметры: DocumentCollection *documents* *documents* Коллекция документов. Возвращаемое значение: нет.	Удаляет заданные документы из базы.

Выгрузка документов

Выгрузка из учетной системы используется, если требуется отправить на терминал определенное задание для работы. В учетной системе, как правило, имеется некоторый документ, содержащий список товаров с количествами, а также, возможно, другие данные (поля шапки и строк документа, табличные части). Например, накладная на приемку. Чтобы реализовать выгрузку документа из учетной системы, нужно определить, какие данные нужны на терминале при работе с документом, т.е. определить состав выгружаемых полей и соответствие полей документа из учетной системы полям документа Mobile SMARTS. Кроме основных полей, объекты Document [Документ] и DocumentItem [СтрокаДокумента] могут содержать произвольные дополнительные поля, доступ к которым осуществляется с помощью функций SetField [УстановитьПоле] и GetField [ПолучитьПоле]. Состав дополнительных полей шапки и строки определяется для каждого типа документа в Панеле управления Mobile SMARTS (см. Тип документа):



Поля шапки



Поля строки

Примечание: некоторые поля, задаваемые в конфигурации Mobile SMARTS являются вычислимыми, т.е. их значение определяется путем некоторых операций над другими полями. Для вычислимого поля задается шаблон значения:

Тип поля	Decimal
Шаблон значения	Item.НоваяЦена > 0? Item.НоваяЦена:Item.Цена

Присваивать значения вычислимых полей при выгрузке нельзя, однако при загрузке получить значение такого поля можно.

Общий алгоритм выгрузки документа. В общем виде алгоритм выгрузки выглядит так:

1. Создаем объект Cleverence.Warehouse.StorageConnector и устанавливаем подключение к нужной базе Mobile SMARTS при помощи вызова SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС];
2. Получаем в учетной системе необходимые для выгрузки документа данные (объект документа учетной системы, некую выборку и т.п.);
3. Создаем объект Document [Документ];
4. Устанавливаем у созданного объекта Document [Документ] свойства:
   - **Id [Ид]** - обычно равен идентификатору или номеру документа в учетной системе. Может быть так, что в учетной системе документы различных типов имеют одинаковый номер и эти разные типы документов выгружаются в Mobile SMARTS. В этом случае Id [Ид] нужно формировать так, чтобы он однозначно определял документ в Mobile SMARTS и для разных выгружаемых документов не мог повторяться (например, <Имя типа документа><разделитель><номер> - “Приемка#БРД000003”);
   - **Name [Имя]** - название документа, которое отображается на терминале при выборе документа из списка и в шапках окон (если не отключено настройками в конфигурации). Можно использовать наименование документа из учетной системы (например, “Приемка БРД000003 от 12.06.16”). Примечание: иногда в списке документов на терминале нужно отобразить дополнительную информацию (например, контрагента, от которого пришел товар), выделить какую-то часть текста в списке цветом и т.п., чтобы пользователю было проще найти нужный документ. В этом случае следует использовать свойство “Шаблон отображения документов в списке” типа документа в Панеле управления, через шаблон можно вывести значения любых основных и дополнительных полей шапки документа (см. Шаблоны);
   - **DocumentTypeName [ИмяТипаДокумента]** - должно быть равно имени одного из типов документов в конфигурации Mobile SMARTS. Например, “Приемка”;
   - **WarehouseId [ИдСклада]** - должен быть равен Ид. одного из складов, имеющихся в конфигурации Mobile SMARTS. В типовых конфигурациях по умолчанию существует единственный склад “Общий” с Ид. равным “1”;
   - **Appointment [Назначение]** - код пользователя или имя группы пользователей, которым

данный документ назначается на исполнение. Может быть пустым (“общий” документ, имеет смысл только для серверной базы Mobile SMARTS). В типовых конфигурациях Mobile SMARTS по умолчанию заведен единственный пользователь с Ид. “оператор”. В самом простом случае присваиваем “оператор”. Если пользователей терминалов несколько, в учетной системе может быть реализован выбор пользователя, которому будет отправлен документ. Для этого нужно получить список пользователей из базы Mobile SMARTS (см. Получение пользователей) и предложить выбор. В этом случае присваивается выбранное значение;

- **AutoAppointed [ВыдаватьАвтоматически]** - по умолчанию Истина (документ сразу отправляется назначенному пользователю). Если нужно, чтобы документ оставался на сервере Mobile SMARTS, пока пользователь на терминале не выберет его из списка или по ШК, свойство следует проставить в Ложь. Например, оставив поле Appointment [Назначение] пустым, а AutoAppointed [ВыдаватьАвтоматически] выставив в Ложь, можно добиться чтобы документ, выгруженный на сервер, был виден всем пользователям терминалов (“общий” документ). Документ остается видным для всех пользователей, пока кто-то один не откроет его на обработку. Имеет смысл только для серверной базы Mobile SMARTS;

5. Проставляем объекту документа необходимые дополнительные поля шапки при помощи SetField [УстановитьПоле]. Состав полей зависит от типа документа. Например, для приемки может потребоваться наименование или идентификатор контрагента, которые берем из исходного документа учетной системы;
6. Обходим в цикле строки исходного документа учетной системы (при необходимости перед этим может быть получена какая-то выборка строк. Например, нужно выгружать не все строки или выполнить группировку). На каждой итерации цикла создаем объект DocumentItem [СтрокаДокумента], в строке документа проставляем поля:
   - **ProductId [ИдТовара]** - должен соответствовать ид. одного из товаров из выгруженного справочника номенклатуры. Для “неизвестного товара” (см. ниже) равно “*”. Если номенклатура не выгружается из учетной системы и реализовано он-лайн получение товара с помощью события сервера ТоварНеНайден (см. События сервера, ТоварНеНайден), функция-обработчик события должна возвращать товар по такому идентификатору;
   - **PackingId [ИдУпаковки]** - упаковка с таким Ид. должна быть в коллекции упаковок данного товара в справочнике товаров базы Mobile SMARTS. Для “неизвестного товара” (см. ниже) равно “шт”. Если номенклатура не выгружается из учетной системы и реализовано он-лайн получение товара с помощью события сервера ТоварНеНайден (см. События сервера, ТоварНеНайден), функция-обработчик события должна возвращать товар, содержащий упаковку с таким идентификатором;
   - **DeclaredQuantity [КоличествоПлан]** - плановое количество товара;
   - При необходимости заполняем поля SSCC [КодЕдиницыХранения], FirstStorageBarcode [ШтрихкодПервогоМеста] и др.
   - Проставляем нужные дополнительные поля строки при помощи SetField [УстановитьПоле]. Например, характеристика товара: documentItem.SetField(“Характеристика”, descr) [строкаДок.УстановитьПоле(“Характеристика”, знач)]
   - **Добавляем** созданную и заполненную строку в коллекцию **DeclaredItems [СтрокиПлан]** документа: document.DeclaredItems.Add(documentItem) [документ.СтрокиПлан.Добавить(строкаДок)];

7. Если кроме заполнения плановых строк требуется выгрузка дополнительных табличных частей документа, подготавливаем данные для выгрузки. Для каждой выгружаемой дополнительной таблицы документа:
   7.1. Создаем объект Table [Таблица], проставляем наименование Name [Имя], как оно задано в конфигурации Mobile SMARTS;
   7.2. Обходим в цикле выгружаемый набор данных (табличную часть документа учетной системы), на каждой итерации цикла создаем объект Row [СтрокаТаблицы];
      - Проставляем поля в строке при помощи SetField [УстановитьПоле], наименования полей должны соответствовать заданным в конфигурации Mobile SMARTS для данной табличной части;
      - Добавляем созданную строку в коллекцию строк таблицы: documentTable.Rows.Add(row) [таблицаДокумента.Строки.Добавить(строка)];
   7.3. Добавляем созданную и заполненную таблицу к таблицам документа: document.Tables.Add(table) [документ.Таблицы.Добавить(таблица)];
8. **Выгружаем** заполненный документ: storageConnector.SetDocument(document) [соединение.ВыгрузитьДокумент(документ)].

“Неизвестный товар”. Строка документа Mobile SMARTS, как правило, определяет конкретный товар с упаковкой (единицей измерения). В этом случае поля ProductId [ИдТовара] и PackingId [ИдУпаковки] соответствуют идентификатору товара и упаковки из справочника номенклатуры. Но возможна ситуация, когда справочник номенклатуры вообще не используется при работе на терминале, а вся необходимая информация содержится в строках документа. Например, в строке документа записан номер рабочего инструмента (без привязки к номенклатуре), а документ представляет собой список всех инструментов, которые необходимо выдать. Тогда номер инструмента записывается в дополнительное поле строки, а при помощи ProductId [ИдТовара] и PackingId [ИдУпаковки] задается “неизвестный товар”. В случае “неизвестного товара” ProductId [ИдТовара] должен быть равен “*”, PackingId [ИдУпаковки] - “шт”.

Пример выгрузки:
C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта соединения

// Выполняем подключение к базе Mobile SMARTS
connector.SelectCurrentApp("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
try
{
   var document = new Cleverence.Warehouse.Document(); //Создаем объект документа Mobile SMARTS
   var acceptanceDoc = AcceptanceJournal.Find(acceptanceDocId); //Получаем документ учетной системы

   document.Id = acceptanceDoc.Id; //Ид. документа
   document.Name = acceptanceDoc.Name; //Имя документа
   document.DocumentTypeName = "Поступление"; //Имя типа документа
   document.WarehouseId = "1"; //Ид. склада, к которому привязан документ

   if(sharedDoc)
   {
      //"Общий" документ - назначение пустое, остается на сервере до явного выбора.
      document.Appointment = "";
      document.AutoAppointed = false;
   }
   else
   {
      document.Appointment = selectedUserId; ///Ид. выбранного пользователя Mobile SMARTS или можно
      //присвоить "оператор" для типовых конфигураций Mobile SMARTS
   }

   //Проставляем дополнительные поля шапки документа
   document.SetField("ИдКонтрагента", acceptanceDoc.Contractor.Id);
   document.SetField("ИдСклада", acceptanceDoc.Warehouse.Id);

   // Обходим в цикле строки табличной части исходного докумнта
   foreach(var acceptanceDocItem in acceptanceDoc.InventItems)
   {
      //Создаем строку документа Mobile SMARTS и присваиваем Ид. товара, Ид. упаковки и плановое //количество
      товара
      var documentItem = new Cleverence.Warehouse.DocumentItem();
      documentItem.ProductId = acceptanceDocItem.InventItem.Id;
      documentItem.PackingId = acceptanceDocItem.Unit.Name;
      documentItem.DeclaredQuantity = acceptanceDocItem.Quantity;

      //Проставляем дополнительные поля строки документа
      if(acceptanceDocItem.InventDim != null)
        documentItem.SetField("Характеристика", acceptanceDocItem.InventDim.Name);

      documentItem.SetField("Цена", acceptanceDocItem.Price);

      //Добавляем строку в коллекцию плановых строк
```

```
        document.DeclaredItems.Add(documentItem);
    }

    //Создание дополнительной таблицы документа
    var boxesTable = new Cleverence.Warehouse.DocumentTable();
    boxesTable.Name = "Короба"; //Присваиваем имя

    foreach(var boxItem in acceptanceDoc.Boxes)
    {
        //Создаем в цикле строки доп. таблицы
        var row = new Cleverence.Warehouse.Row();
        row.SetField("Номер", boxItem.Id);
        row.SetField("Описание", boxItem.Description);
        boxesTable.Rows.Add(row); //Добавление строки в коллекцию строк таблицы
    }

    //Добавляем таблицу в коллекцию таблиц документа
    document.Tables.Add(boxesTable);

    connector.SetDocument(document); //Выгружаем документ

}
catch
{
  //Обработка ошибки
}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
//соединения
Попытка
    connector.УстановитьПодключениеСБазойСМАРТС("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
  //Выполняем подключение
  ДокументТСД = новый COMОбъект("Cleverence.Warehouse.Document");
  Документ1С = Документы.ПоступлениеТоваров.НайтиПоНомеру(НомерДокумента);

  ДокументТСД.Ид = “Поступление#”+Документ1С.Номер; //Ид. документа
  ДокументТСД.Имя = Строка(Документ1С); //Имя документа
  ДокументТСД.ИмяТипаДокумента = "Поступление"; //Имя типа документа
  ДокументТСД.ИдСклада = "1"; //Ид. склада, к которому привязан документ

  Если ОбщийДокумент = Истина Тогда
     //"Общий" документ - назначение пустое, остается на сервере до явного выбора.
     ДокументТСД.Назначение = "";
     ДокументТСД.ВыдаватьАвтоматически = Ложь;
  Иначе
     ДокументТСД.Назначение = ИдПользователяТСД; //Ид. выбранного пользователя Mobile SMARTS или можно
     //присвоить "оператор" для типовых конфигураций Mobile SMARTS
  КонецЕсли;

  //Проставляем дополнительные поля шапки документа
  ДокументТСД.УстановитьПоле("ИдКонтрагента", ДокументТСД.Контрагент.Код);
  ДокументТСД.УстановитьПоле("ИдСклада", ДокументТСД.Склад.Код);

  Для каждого строкаДок1С из Документ1С.Товары Цикл
      //Создаем строку документа Mobile SMARTS и присваиваем Ид. товара, Ид. упаковки и плановое //количество
      товара
     СтрокаДокументаТСД = Новый COMОбъект("Cleverence.Warehouse.DocumentItem");
     СтрокаДокументаТСД.ИдТовара = XMLСтрока(строкаДок1С.Номенклатура);
```

```
    СтрокаДокументаТСД.ИдУпаковки = строкаДок1С.Упаковка.Наименование;
    СтрокаДокументаТСД.КоличествоПлан = строкаДок1С.Количество;

    //Проставляем дополнительные поля строки документа
    СтрокаДокументаТСД.УстановитьПоле("Характеристика", Строка(строкаДок1С.Характеристика));
    СтрокаДокументаТСД.УстановитьПоле("Цена", строкаДок1С.Цена);

    //Добавляем строку в коллекцию плановых строк
    ДокументТСД.СтрокиПлан.Добавить(СтрокаДокументаТСД);
КонецЦикла;

//Создание дополнительной таблицы документа
ТаблицаКоробовТСД = Новый COMОбъект("Cleverence.Warehouse.Table");
ТаблицаКоробовТСД.Имя = "Короба";

Для каждого строкаКоробаДок1С из Документ1С.Короба Цикл
    //Создаем в цикле строки доп. таблицы
    СтрокаКоробаТСД = Новый COMОбъект("Cleverence.Warehouse.Row");
    СтрокаКоробаТСД.УстановитьПоле("Номер", строкаКоробаДок1С.НомерКороба);
    СтрокаКоробаТСД.УстановитьПоле("Описание", строкаКоробаДок1С.Описание);

    ТаблицаКоробовТСД.Строки.Добавить(СтрокаКоробаТСД); //Добавление строки в коллекцию строк таблицы
КонецЦикла;

//Добавляем таблицу в коллекцию таблиц документа
ДокументТСД.Таблицы.Добавить(ТаблицаКоробовТСД);

connector.ВыгрузитьДокумент(ДокументТСД); //Выгружаем документ

Исключение
// При выгрузке возникли ошибки, обрабатываем исключение

КонецПопытки;
```

Загрузка документов

Для того, чтобы отразить результаты работы пользователей терминалов в учетной системе, следует загрузить завершенные документы, содержащие фактические данные. Например, после приемки товара, кладовщик завершает документ на терминале, в документе содержится список фактически принятых товаров с количествами, этот список загружается в учетную систему и на его основе заполняется табличная часть документа учетной системы, после чего документ проводится и в учетной системе фиксируется принятый товар.

Общий алгоритм загрузки документов.

1. Создаем объект Cleverence.Warehouse.StorageConnector и устанавливаем подключение к нужной базе Mobile SMARTS при помощи вызова SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС];
2. Получаем из базы Mobile SMARTS документы для загрузки. Если нужно загрузить все завершенные документы (возможно, с фильтром по типу документа), используем функцию GetDocuments [ПолучитьДокументы]. Если известен идентификатор документа, который требуется загрузить, получаем один этот документ с помощью функции GetDocument [ПолучитьДокумент];
3. Для загрузки данных из документа Mobile SMARTS может быть создан новый документ учетной системы или загрузка может выполняться в существующий. Если загружается документ ранее выгруженный из учетной системы, загрузка может выполняться в исходный документ (при этом заполняется фактический список товаров и количества) или может быть создан новый документ другого типа (например, выгружали Заказ поставщику, при загрузке создаем Поступление товаров), также пользователь учетной системы может выбрать документ, в который будет происходить загрузка. Конкретный вариант определяется принятым бизнес-процессом. Перед загрузкой в существующий документ нужно обнулить количества товара в табличной части документа, в который происходит загрузка, или очистить табличную часть (Если для количества используется

только одно поле и нет двух полей, к примеру, Количество и КоличествоФакт).

4. Получаем из документа Mobile SMARTS поля шапки, которые требуются, с помощью вызова GetField [ПолучитьПоле], и записываем нужные данные в поля документа учетной системы;
5. Обходим в цикле строки фактической части документа Mobile SMARTS CurrentItems [СтрокиФакт], на каждой итерации цикла:
   5.1. На основании значений полей ProductId [ИдТовара], PackingId [ИдУпаковки] строки документа Mobile SMARTS находим в учетной системе товар и упаковку (единицу измерения);
   5.2. Получаем из строки документа Mobile SMARTS необходимые при загрузке дополнительные поля при помощи GetField [ПолучитьПоле] (например, Характеристика, Цена);
   5.3. Ищем в табличной части документа учетной системы строку с данным товаром и другими соответствующими полями (например, характеристикой). Если строка найдена, прибавляем количество товара. Если нет - создаем новую строку, записываем в нее товар, упаковку, другие поля если нужно (например, характеристику, цену);
6. Когда все данные загружены, записываем документ в учетной системе;
7. Если загруженный документ Mobile SMARTS больше не нужен, удаляем его из базы при помощи функции RemoveDocument [УдалитьДокумент].

Пример загрузки:
C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта соединения

// Выполняем подключение к базе Mobile SMARTS
connector.SelectCurrentApp("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
try
{
   // Получаем из базы все завершенные документы поступления
   var documents = connector.GetDocuments("Поступление", true);

   //Обходим в цикле полученные документы и обрабатываем каждый документ
   foreach(var document in documents)
   {
      var acceptanceDoc = AcceptanceJournal.Find(document.Id); //Ищем в учетной системе исходный //документ
      приемки, который был выгружен в Mobile SMARTS
      if(acceptanceDoc == null) //Если документ не найден, создаем новый
         acceptanceDoc = AcceptanceJournal.CreateDocument();
      else
         acceptanceDoc.InventItems.Clear(); //Очистим строки, чтобы загрузить фактически набранный //товар

      //Получаем поля шапки документа Mobile SMARTS
      string contractorId = document.GetField("ИдКонтрагента") as string;
      string warehouseId = document.GetField("ИдСклада") as string;

      //На основании полей шапки документа Mobile SMARTS заполняем поля шапки документа учетной системы
      if(!String.IsNullOrEmpty(contractorId))
      {
         acceptanceDoc.Contractor = ContractorCatalog.FindById(contractorId);
      }
      if(!String.IsNullOrEmpty(warehouseId))
      {
         acceptanceDoc.Warehouse = WarehouseCatalog.FindById(warehouseId);
      }

      //Обходим в цикле фактические строки документа Mobile SMARTS
      foreach(var documentItem in document.CurrentItems)
      {
         //Получаем из строки документа Mobile SMARTS Ид. товара, Ид. упаковки и количество
         string productId = documentItem.ProductId;
         string packingId = documentItem.PackingId;
         decimal currentQuantity = documentItem.CurrentQuantity;
```

```
        //Получаем дополнительные поля строки документа
        string descr = documentItem.GetField("Характеристика");
        decimal price = documentItem.GetField("Цена");

        //Находим в учетной системе товар, единицу и характеристику
        var inventItem = InventItemCatalog.FindById(productId);
        var unit = UnitCatalog.FindByName(packingId);

        var inventDim = (from inventDim in InventDimCatalog
           where inventDim.Owner == inventItem && inventDim.Name == descr
           select inventDim).FirstOrDefault();

        //Ищем строку документа учетной системы с полученными товаром, единицей и характеристикой
        var acceptanceDocItem = (from docItem in acceptanceDoc.InventItems
           where docItem.InventItem == inventItem && docItem.Unit == unit
           && docItem.InventDim == inventDim
           select docItem).FirstOrDefault();

        if(acceptanceDocItem != null)
        {
           //Если строка найдена, прибавляем количество
           acceptanceDocItem.Quantity += currentQuantity;
        }
        else
        {
           //Строка не найдена, добавляем строку и заполняем в ней нужные поля
           acceptanceDocItem = acceptanceDoc.InventItems.AddNew();
           acceptanceDocItem.InventItem = inventItem;
           acceptanceDocItem.Unit = unit;
           acceptanceDocItem.InventDim = inventDim;
           acceptanceDocItem.Price = price;
           acceptanceDocItem.Quantity = currentQuantity;
        }
     }

     //Все строки загрузили, сохраняем документ
     acceptanceDoc.Save();

     //Удаляем из базы Mobile SMARTS загруженный документ
     connector.RemoveDocument(document.Id);
  }
}
catch
{
  //Обработка ошибки
}
```

«1С:Предприятие 8»:

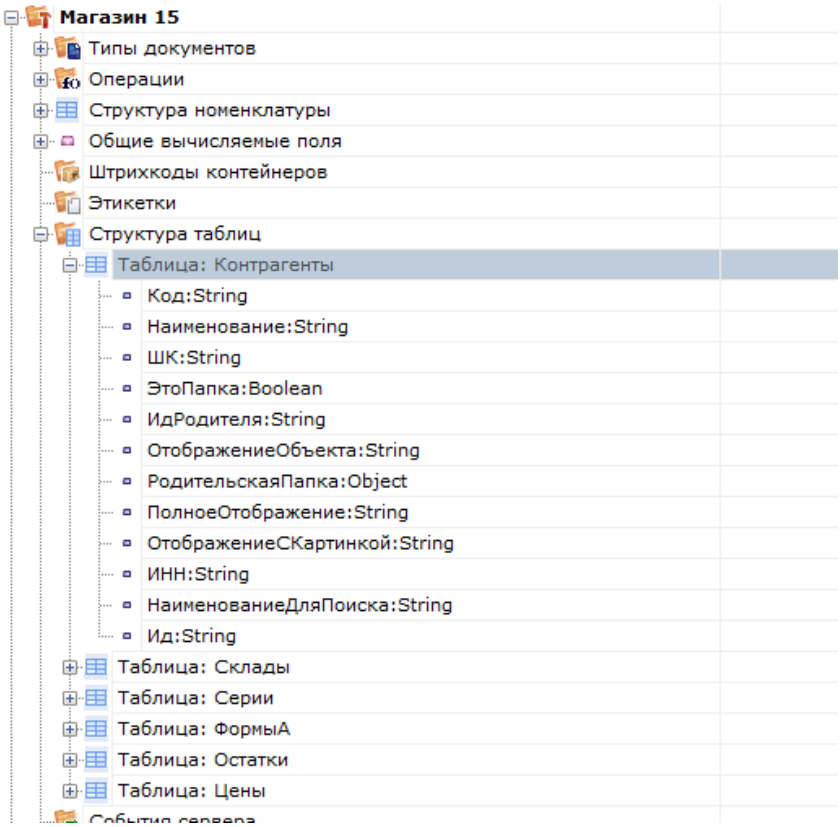
```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
//соединения
Попытка
    connector.УстановитьПодключениеСБазойСМАРТС("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
  //Выполняем подключение

  // Получаем из базы все завершенные документы поступления
  ДокументыТСД = connector.ПолучитьДокументы("Поступление", Истина);

  //Обходим в цикле полученные документы и обрабатываем каждый документ
```

```
Для ИндДок = 0 По ДокументыТСД.Количество-1 Цикл
   ДокументТСД = ДокументыТСД.Элемент(ИндДок);
   Документ1С = НайтиДокумент1СПоИдТСД(ДокументТСД.Ид); //Получаем документ 1С по ид. документа ТСД. //При
   выгрузке из 1С назначали ид. вида “Поступление#”+Документ1С.Номер, для созданного на //терминале
   документ 1С = Неопределено, создаем новый документ 1С

   Если Документ1С = Неопределено Тогда
      Документ1С = Документы.ПоступлениеТоваров.СоздатьДокумент();
   Иначе
      Документ1С.Товары.Очистить();
   КонецЕсли;

   //Получаем поля шапки документа Mobile SMARTS
   КодКонтрагента = ДокументТСД.ПолучитьПоле("ИдКонтрагента");
   КодСклада = ДокументТСД.ПолучитьПоле("ИдСклада");

   //На основании полей шапки документа Mobile SMARTS заполняем поля шапки документа 1С
   Если ЗначениеЗаполнено(КодКонтрагента) Тогда
      Документ1С.Контрагент = Справочники.Контрагенты.НайтиПоКоду(КодКонтрагента);
   КонецЕсли;

   Если ЗначениеЗаполнено(КодСклада) Тогда
      Документ1С.Склад = Справочники.Склады.НайтиПоКоду(КодСклада);
   КонецЕсли;

   //Обходим в цикле фактические строки документа Mobile SMARTS
   Для ИндСтроки = 0 По ДокументТСД.СтрокиФакт.Количество-1 Цикл
      СтрокаДокументаТСД = ДокументТСД.СтрокиФакт.Элемент(ИндСтроки); //Получаем строку документа

      //Получаем из строки документа Mobile SMARTS Ид. товара, Ид. упаковки и количество
      ИдТовара = СтрокаДокументаТСД.ИдТовара;
      ИмяУпаковки = СтрокаДокументаТСД.ИдУпаковки;
      Количество = СтрокаДокументаТСД.КоличествоФакт;

      //Получаем дополнительные поля строки документа
      ХарактеристикаИмя = СтрокаДокументаТСД.ПолучитьПоле(“Характеристика”);
      Цена = СтрокаДокументаТСД.ПолучитьПоле(“Цена”);

      //Находим в 1С товар, единицу и характеристику
      Номенклатура = Справочники.Номенклатура.ПолучитьСсылку(новый УникальныйИдентификатор(ИдТовара ));

      Упаковка = Справочники.ЕдиницыИзмерения.НайтиПоНаименованию(ИмяУпаковки,,,Номенклатура);

      Если ЗначениеЗаполнено(ХарактеристикаИмя) Тогда
         Характеристика = Справочники.Характеристики.НайтиПоНаименованию(ХарактеристикаИмя ,,,Номенклатура);
      КонецЕсли;

       //Ищем строку документа 1С с полученными товаром, упаковкой и характеристикой
      ПараметрыОтбора = Новый Структура;
      ПараметрыОтбора.Вставить("Номенклатура", Номенклатура);
      ПараметрыОтбора.Вставить("Упаковка", Упаковка);
      ПараметрыОтбора.Вставить("Характеристика", Характеристика);

      НайденныеСтроки = Документ1С.Товары.НайтиСтроки(ПараметрыОтбора);

      Если НайденныеСтроки.Количество() > 0 Тогда
         СтрокаТабличнойЧасти = НайденныеСтроки[0];
         //Если строка найдена, прибавляем количество
         СтрокаТабличнойЧасти.Количество = СтрокаТабличнойЧасти.Количество + Количество;
      Иначе
```

```
            //Строка не найдена, добавляем строку и заполняем в ней нужные поля
            СтрокаТабличнойЧасти = Документ1С.Товары.Добавить();
            СтрокаТабличнойЧасти.Номенклатура = Номенклатура;
            СтрокаТабличнойЧасти.Упаковка = Упаковка;
            СтрокаТабличнойЧасти.Характеристика = Характеристика;
            СтрокаТабличнойЧасти.Количество = Количество;
            СтрокаТабличнойЧасти.Цена = Цена;
        КонецЕсли;

        //Записываем документ в 1С
        Документ1С.Записать(РежимЗаписиДокумента.Проведение);

        //Удаляем загруженный документ Mobile SMARTS из базы
        connector.УдалитьДокумент(ДокументТСД.Ид);
    КонецЦикла;

  КонецЦикла;

Исключение
// Обработка ошибки

КонецПопытки;
```

Дополнительные таблицы

Дополнительная таблица представляет собой “плоский” набор данных, строки таблицы содержат поля простых типов (числа, строки, булевы). Структура данных дополнительных таблиц создается и редактируется с помощью Панели управления Mobile SMARTS. Разработчик конфигурации может создавать произвольные дополнительные таблицы. В таблице кроме полей, которые содержат данные, могут быть заданы вычислимые поля (значения вычислимых полей определяется путем операций над значениями других полей), для вычислимых полей задается Шаблон значения. Таблица может как загружаться на терминал, так и хранится на сервере (в случае серверной базы Mobile SMARTS).
Например, таблица Контрагенты используется для хранения списка сторонних организаций, на терминале при работе с документом Приемки пользователь может выбирать организацию, от которой пришел товар. В таблице Остатки хранятся остатки товаров в разрезе складов, привязка к номенклатуре выполняется с помощью поля ИдНоменклатуры.



Дополнительные таблицы

Для работы с дополнительными таблицами используется объект TableAccessor [ДоступКТаблице], с помощью функций этого объекта можно выгружать данные в таблицу, читать данные, добавлять и удалять строки таблицы. Для получения объекта TableAccessor [ДоступКТаблице] следует использовать функцию GetTableAccessor [ПолучитьДоступКТаблице] из StorageConnector [Соединение]. Перед вызовом GetTableAccessor [ПолучитьДоступКТаблице] должно быть установлено подключение к базе данных Mobile SMARTS с помощью SelectCurrentApp [УстановитьПодключениеСБазойСМАРТС].

Наименование	**Параметры и возвращаемое значение**	**Описание**
GetTableAccessor [ПолучитьДоступКТаблице]	string *tableName* Имя таблицы, для которой нужно получить объект доступа. Должно соответствовать имени одной из таблиц из конфигурации Mobile SMARTS. Возвращаемое значение: объект TableAccessor [ДоступКТаблице].	Возвращает объект TableAccessor [ДоступКТаблице], позволяющий выгружать данные в таблицу, читать данные, добавлять и удалять строки таблицы.

Объект TableAccessor [ДоступКТаблице] имеет следующие функции:

Наименование	**Параметры и возвращаемое значение**	**Описание**
Выгрузка		
BeginUpload [НачатьВыгрузку]	*bool overwrite* Истина - полностью перезаписать таблицу при выгрузке, Ложь - добавлять	Открывает выгрузку данных в таблицу.

	строки Возвращаемое значение: нет	
Upload [ДобавитьВВыгрузку]	*RowCollection rows* rows - коллекция выгружаемых строк Возвращаемое значение: нет	Добавляет порцию строк в выгрузку.
EndUpload [ЗавершитьВыгрузку]	Параметры: нет Возвращаемое значение: нет	Завершает выгрузку. Данные сохраняются в базу Mobile SMARTS.
ResetUpload [СброситьВыгрузку]	Параметры: нет Возвращаемое значение: нет	Сбрасывает начатую выгрузку без сохранения данных.
Чтение данных **Свойства:**		
Count [Количество]	Возвращаемое значение: Число	Количество строк в таблице.
Item [Элемент]	int *index* index - индекс строки Возвращаемое значение: нет	Получает строку по индексу.
Поиск		
FindByField [НайтиПоЗначениюПоля]	*string fieldName,* *object value* *fieldName* Наименование поля для отбора. *value* Значение для отбора Возвращаемое значение: TableAccessor [ДоступКТаблице]	Выполняет поиск в таблице по значению заданного поля. Возвращает объект TableAccessor [ДоступКТаблице], из которого можно читать строки, удовлетворяющие отбору.
FindFirstByField [НайтиПервуюЗаписьПоЗначениюПоля]	*string fieldName,* *object value* *fieldName* Наименование поля для отбора. *value* Значение для отбора Возвращаемое значение: Row Найденная строка таблицы или null.	Выполняет поиск в таблице по значению заданного поля и возвращает первую найденную строку или null, если строка не найдена.
Редактирование		
Add [Добавить]	Row *item* *item* - добавляемая строка Возвращаемое значение: нет	Добавляет строку в таблицу. Чтобы зафиксировать сделанные изменения, следует вызвать CommitChanges [СохранитьИзменения].
Remove [Удалить]	Row *item*	Удаляет строку из таблицы.

	item - строка таблицы для удаления Возвращаемое значение: bool	Чтобы зафиксировать сделанные изменения, следует вызвать CommitChanges [СохранитьИзменения]. .
CommitChanges [СохранитьИзменения]	Параметры: нет Возвращаемое значение: нет	Сохраняет сделанные изменения в таблицу в базе данных Mobile SMARTS.
UndoChanges [ОтменитьИзменения]	Параметры: нет Возвращаемое значение: нет	Отменяет сделанные изменения (добавленные, удаленные строки).

Пример выгрузки:
C#:

```
var connector = new Cleverence.Warehouse.StorageConnector(); // Создаем экземпляр объекта соединения

// Выполняем подключение к базе Mobile SMARTS
connector.SelectCurrentApp("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
// Получаем объект для доступа к таблице
var tableContractors = connector.GetTableAccessor("Контрагенты");

try
{
   tableContractors.BeginUpload(true); // Начинаем выгрузку

   var rows = new Cleverence.Warehouse.RowCollection(); // Коллекция строк таблицы, которую будем
   // использовать при выгрузке

  using (SqlConnection connection = new SqlConnection(connectionString)) //Устанавливаем соединение с //БД,
  из которой будем читать данные
   {
     connection.Open();
     command.Connection = connection;
     SqlDataReader reader = command.ExecuteReader();
      while (reader.Read()) //Читаем данные
      {
        string id = reader.GetString(0);
        string name = reader.GetString(1);
        string inn = reader.GetString(1);

        if(rows.Count >= 1000)
        {
           //Если набрали достаточно строк, выгружаем порцию строк и создаем новый объект коллекции,
           //чтобы набирать строки дальше в пустую коллекцию.
           tableContractors.Upload(rows);
           rows = new Cleverence.Warehouse.RowCollection();
        }

        var row = new Cleverence.Warehouse.Row(); //Создаем строку таблицы
        row.SetField("Ид", id); //Проставляем поля
        row.SetField("Наименование", name);
        row.SetField("ИНН", inn);

        rows.Add(row); //Добавляем строку в коллекцию строк
      }

     //Если есть строки для выгрузки, выгружаем
     if(rows.Count > 0)
```

```
        tableContractors.Upload(rows);
    }

    tableContractors.EndUpload(); //Завершаем выгрузку
}
catch
{
    //Обработка ошибки
    tableContractors.ResetUpload(); //Если возникла ошибка, сбрасываем выгрузку

}
```

«1С:Предприятие 8»:

```
connector = новый COMОбъект("Cleverence.Warehouse.StorageConnector"); // Создаем экземпляр объекта
//соединения
connector.УстановитьПодключениеСБазойСМАРТС("михаил-пк:10501/d7c15f54-fd05-4811-8f6d-c7c7003887c5");
//Выполняем подключение

таблицаКонтрагенты = connector.ПолучитьДоступКТаблице("Контрагенты"); // Получаем объект для доступа к //
таблице

Попытка
  таблицаКонтрагенты.НачатьВыгрузку(Истина); // Начинаем выгрузку
строкиДляВыгрузки = новый COMОбъект("Cleverence.Warehouse.RowCollection"); // Коллекция строк //таблицы,
которую будем использовать при выгрузке

  Запрос = Новый Запрос(ТекстЗапроса);
  ТаблицаРезультата = Запрос.Выполнить().Выгрузить(); //Выполняем запрос

  Для каждого СтрокаТаблицы из ТаблицаРезультата Цикл
     Ид = XMLСтрока(СтрокаТаблицы.КонтрагентСсылка); //Получаем необходимые поля
     Наименование = СтрокаТаблицы.Наименование;
     ИНН = СтрокаТаблицы.ИНН;

     Если строкиДляВыгрузки.Количество >= 1000 Тогда
        //Если набрали достаточно строк, выгружаем порцию и очищаем коллекцию строк
        таблицаКонтрагенты.ДобавитьВВыгрузку(строкиДляВыгрузки);
        таблицаКонтрагенты.Очистить();
     КонецЕсли;

     строкаДляВыгрузки = новый COMОбъект("Cleverence.Warehouse.Row"); //Создаем строку таблицы
     строкаДляВыгрузки.УстановитьПоле(“Ид”, Ид); //Проставляем поля
     строкаДляВыгрузки.УстановитьПоле(“Наименование”, Наименование);
     строкаДляВыгрузки.УстановитьПоле(“ИНН”, ИНН);

     строкиДляВыгрузки.Добавить(строкаДляВыгрузки); //Добавляем строку в коллекцию строк
  КонецЦикла;

  //Если есть строки для выгрузки, выгружаем
  Если строкиДляВыгрузки.Количество > 0 Тогда
     таблицаКонтрагенты.ДобавитьВВыгрузку(строкиДляВыгрузки);
  КонецЕсли;

  таблицаКонтрагенты.ЗавершитьВыгрузку(); //Завершаем выгрузку
Исключение
  // Обработка ошибки
  таблицаКонтрагенты.СброситьВыгрузку(); //Если возникла ошибка, сбрасываем выгрузку
КонецПопытки;
```

Работа с компонентой AddIn.Cl.TerminalConnector

Компонента AddIn.Cl.TerminalConnector предназначена для использования в среде “1С:Предприятие” (версий 7.7, 8.x). Компонента реализует стандарт 1С на внешние компоненты, а также стандарты 1С на компоненты драйверов торгового оборудования (терминалов сбора данных). Компонента позволяет выполнять соединение с базой данных Mobile SMARTS, выгружать справочник номенклатуры и дополнительные таблицы, выгружать документы Mobile SMARTS для исполнения на терминале, загружать завершенные документы Mobile SMARTS в 1С. Отличие от Cleverence.Warehouse.StorageConnector заключается в том, что для обмена данными используются объекты 1С (массивы, списки значений, таблицы значений и др.), что существенно упрощает написание кода обмена в 1С. При этом возможно совместное использование Add.Cl.TerminalConnector и Cleverence.Warehouse.StorageConnector, а также всех других COM-объектов Mobile SMARTS (Document [Документ], DocumentItem [СтрокаДокумента], Product [Товар] и др.).

Подключение компоненты и создание объекта:

«1С:Предприятие 8»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
Если ПодключитьВнешнююКомпоненту(ПрогИД) Тогда
   КомДляMS = Новый (ПрогИД);
Иначе
   //Ошибка подключения компоненты
КонецЕсли;
```

«1С:Предприятие 7.7»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
Если ПодключитьВнешнююКомпоненту(ПрогИД) <> 0 Тогда
   ОбъектТСД = СоздатьОбъект(Компонента);
Иначе
   //Ошибка подключения компоненты
КонецЕсли;
```

Если по каким-то причинам использование ПодключитьВнешнююКомпоненту с передачей программного идентификатора невозможно (например, на сервере “1С:Предприятие”), компонента может быть создана как COM-объект:

«1С:Предприятие 8»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
КомДляMS = Новый COMОбъект(ПрогИД);
КомДляMS.УстановитьВерсию1С("v8"); //Необходимо сообщить компоненте версию платформы 1С (“v7” или “v8”)
```

«1С:Предприятие 7.7»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
КомДляMS = СоздатьОбъект(ПрогИД);
КомДляMS.УстановитьВерсию1С("v7"); //Необходимо сообщить компоненте версию платформы 1С (“v7” или “v8”)
```

Подключение к базе данных Mobile SMARTS

Перед выполнением обмена данными (выгрузка номенклатуры, документов и др.) необходимо установить подключение к базе данных Mobile SMARTS, для этого используется функция Open [Подключить].

Наименование	Параметры и возвращаемое значение	Описание
Open [Подключить]	Параметры: *ValuesArray [МассивЗначений],*	Выполняет подключение к базе данных Mobile SMARTS и

	DeviceID [ИДУстройства] *ValuesArray [МассивЗначений]* - Массив (для 1С 8.x) или Список значений (для 1С 7.7), содержащий один элемент, в котором должен находится идентификатор базы Mobile SMARTS (или строка подключения), к которой происходит подключение. *DeviceID [ИДУстройства]* - Возвращается идентификатор ТСД или пустая строка. Возвращаемое значение: Истина (1) - подключение успешно выполнено, Ложь (0) - возникла ошибка. Для получения описания ошибки используйте функцию GetLastError [ПолучитьОшибку].	инициализацию компоненты драйвера для последующего обмена с терминалом, папкой обмена или сервером Mobile SMARTS. Функция входит в один из стандартов 1С на Драйвер для ТСД (См. Описание стандарта).

«1С:Предприятие 8»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
Если ПодключитьВнешнююКомпоненту(ПрогИД) Тогда
   КомДляMS = Новый (ПрогИД);

   МассивЗначений = Новый Массив;
   МассивЗначений.Добавить(“михаил-пк:10501/3cb8cb43-ca60-42c5-94d9-bb352dcba9e1”); //Строка
   // подключения или ид. базы

   ИДУстройства = “”;
   Если КомДляMS.Подключить(МассивЗначений, ИДУстройства) Тогда
      //Успешное подключение
   Иначе
      //Ошибка при подключении к базе
        ОписаниеОшибки = “”;
        КомДляMS.ПолучитьОшибку(ОписаниеОшибки);
   КонецЕсли;

Иначе
   //Ошибка подключения компоненты
КонецЕсли;
```

«1С:Предприятие 7.7»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
Если ПодключитьВнешнююКомпоненту(ПрогИД) <> 0 Тогда
   КомДляMS = СоздатьОбъект(ПрогИД);

   Параметры = СоздатьОбъект("СписокЗначений");
   Параметры.ДобавитьЗначение(“михаил-пк:10501/3cb8cb43-ca60-42c5-94d9-bb352dcba9e1”); //Строка
   // подключения или ид. базы

   ИДУстройства = “”;
   Если КомДляMS.Подключить(Параметры, ИДУстройства) <> 0 Тогда
        //Успешное подключение
   Иначе
        //Ошибка при подключении к базе
        ОписаниеОшибки = “”;
```

```
        КомДляMS.ПолучитьОшибку(ОписаниеОшибки);
    КонецЕсли;

Иначе
    //Ошибка подключения компоненты
КонецЕсли;
```

Выгрузка номенклатуры

Для того, чтобы реализовать выгрузку справочника номенклатуры из 1С в Mobile SMARTS нужно определить, какие данные потребуются на терминале для работы и как отобразятся эти данные на объекты Mobile SMARTS Product [Товар] и Packing [Упаковка] при выгрузке. См. также Выгрузка номенклатуры. Самое очевидное, что должны выгружаться штрихкоды товаров и наименования (для того, чтобы при сканировании на терминале происходила идентификация товара по штрихкоду). Штрихкод в базе 1С, как правило, привязан не только к позиции номенклатуры, но и к упаковке (единице измерения), а часто и к характеристике номенклатуры. Таким образом, позиции номенклатуры соответствует объект Product [Товар], упаковкам и характеристикам - Packing [Упаковка]. Наименование характеристики записывается в дополнительное поле Характеристика (descr) упаковки Packing [Упаковка]. Для объектов Product [Товар] требуются уникальные идентификаторы (поле Id), лучше всего использовать Guid-ы позиций справочника номенклатуры (для 1С 7.7 - коды). Кроме того, при выгрузке могут заполняться различные дополнительные поля номенклатуры, определенные в конфигурации Mobile SMARTS. Например, Колво (qty) - остаток товара на складе, Алко - признак, что товар является алкоголем и на терминале следует сканировать ШК акцизной марки. Когда определены все нужные для выгрузки данные, можно приступить к написанию запроса для их получения. Чаще всего запрос выполняется к справочнику номенклатуры и используются соединения для регистров штрихкодов, остатков, цен и др. Товары, не имеющие штрихкодов, тоже обычно выгружаются в Mobile SMARTS, т.к. могут выгружаться документы, содержащие такие товары и чтобы видеть на терминале наименование и другие данные по товару, нужно, чтобы он присутствовал в справочнике номенклатуры. Кроме того, штрихкод к товару можно присвоить на самом терминале.

Для выгрузки используются следующие функции объекта AddIn.Cl.TerminalConnector:

Наименование	**Параметры и возвращаемое значение**	**Описание**
BeginUploadProducts [НачатьВыгрузкуТоваров]	Параметры: *schema [СхемаВыгрузки]* Массив (для 1С 8.x) или Список значений (для 1С 7.7), описывающий схему выгрузки. Содержит наименования полей объектов Product [Товар] и Packing [Упаковка] в виде “Product.ИмяПоля” или “Packing.ИмяПоля”. Значения полей передаются с помощью функции AddProductToUpload [ДобавитьВВыгрузкуТоваров]. Возвращаемое значение: Истина (1) - выполнено успешно, Ложь (0) - возникла ошибка. Для получения описания ошибки используйте функцию GetLastError [ПолучитьОшибку].	Переводит коннектор в режим выгрузки товаров. Выгрузка выполняется с помощью функции AddProductToUpload [ДобавитьВВыгрузкуТоваров].

AddProductToUpload [ДобавитьВВыгрузкуТоваров]	Параметры: *object row [Данные]* Массив (для 1С 8.x) или Список значений (для 1С 7.7), содержащий поля объектов Product [Товар] и Packing [Упаковка], заданные при помощи функции BeginUploadProducts [НачатьВыгрузкуТоваров]. Возвращаемое значение: Истина (1) - выполнено успешно, Ложь (0) - возникла ошибка. Для получения описания ошибки используйте функцию GetLastError [ПолучитьОшибку].	Добавляет в выгрузку товаров товар с упаковкой.
EndUploadProducts [ЗавершитьВыгрузкуТоваров]	Параметры: нет Возвращаемое значение: Истина (1) - выполнено успешно, Ложь (0) - возникла ошибка. Для получения описания ошибки используйте функцию GetLastError [ПолучитьОшибку].	Завершает выгрузку товаров.
ResetUploadProducts [СброситьВыгрузкуТоваров]	Параметры: нет Возвращаемое значение: Истина (1) - выполнено успешно, Ложь (0) - возникла ошибка. Для получения описания ошибки используйте функцию GetLastError [ПолучитьОшибку].	Сбрасывает начатую выгрузку. Функция применяется в случае ошибки в процессе выгрузки номенклатуры.

«1С:Предприятие 8»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
ПодключитьВнешнююКомпоненту(ПрогИД);
Соединение = Новый (ПрогИД);

МассивЗначений = Новый Массив;
МассивЗначений.Добавить(“михаил-пк:10501/3cb8cb43-ca60-42c5-94d9-bb352dcba9e1”); //Строка
// подключения или ид. базы

ИДУстройства = “”;
Если КомДляMS.Подключить(МассивЗначений, ИДУстройства) Тогда
  Запрос = Новый Запрос(ТекстЗапроса);
  ТаблицаТоваров = Запрос.Выполнить().Выгрузить(); //Выполняем запрос

 //Описываем схему выгрузки
  мДанные = Новый Массив(26);
 //Основные поля товара и упаковки
  мДанные.Установить(  0, "Product.Id" );
  мДанные.Установить(  1, "Product.Marking" );
  мДанные.Установить(  2, "Product.Barcode" );
  мДанные.Установить(  3, "Packing.Barcode" );
  мДанные.Установить(  4, "Product.Name" );
```

```
мДанные.Установить(  5, "Product.BasePackingId" );
мДанные.Установить(  6, "Packing.Id" );
мДанные.Установить(  7, "Packing.Name" );
мДанные.Установить(  8, "Packing.UnitsQuantity" );
//Дополнительные поля
мДанные.Установить(  9, "Packing.descr" );
мДанные.Установить( 10, "Packing.serial" );
мДанные.Установить( 11, "Packing.price" );
мДанные.Установить( 12, "Packing.qty" );
мДанные.Установить( 13, "Product.withserial" );
мДанные.Установить( 14, "Product.withsn" );
мДанные.Установить( 15, "Product.КлючСерий" );
//Дополнительные поля для работы с алкоголем
мДанные.Установить( 16, "Product.Алко" );
мДанные.Установить( 17, "Packing.АлкоОбъем" );
мДанные.Установить( 18, "Packing.АлкоКодВ" );
мДанные.Установить( 19, "Packing.АлкоНаимВ" );
мДанные.Установить( 20, "Product.АлкоКрепость" );
мДанные.Установить( 21, "Product.Производитель" );
мДанные.Установить( 22, "Product.ПроизвИНН" );
мДанные.Установить( 23, "Product.ПроизвКПП" );
мДанные.Установить( 24, "Packing.АлкоКод" );
мДанные.Установить( 25, "Packing.АлкоМарк" );

Соединение.УстановитьПоискПоНаименованиюИАртикулу(Истина); //Устанавливаем признак, что будет //доступен
поиск по наименованию товара на терминале

//Начинаем выгрузку
Если Не Соединение.НачатьВыгрузкуТоваров(мДанные) Тогда
//Ошибка при начале выгрузки
      ОписаниеОшибки = “”;
      Соединение.ПолучитьОшибку(ОписаниеОшибки);
      Соединение.ОсвободитьРесурсы();

      Сообщить(ОписаниеОшибки);
      Возврат;
КонецЕсли;

Для Каждого СтрокаТовара из ТаблицаТоваров Цикл
   ИдТовара = XMLСтрока(СтрокаТовара[“НоменклатураСсылка”]); //В качестве идентификатора товара //будем
   выгружать Guid позиции номенклатуры
   Наименование              = Строка(СтрокаТовара["Номенклатура"]);
   Код       = СокрЛП(СтрокаТовара["Код"]);
   ШК  = СокрЛП(СтрокаТовара["Штрихкод"]);
   ЕдиницаИзмерения          = Строка(СтрокаТовара["Упаковка"]);
   Характеристика           = СокрЛП(СтрокаТовара["Характеристика"]);
   Серия             = СокрЛП(СтрокаТовара["Серия"]);
   Цена              = СтрокаТовара["Цена"];
   Количество        = СтрокаТовара["Количество"];
   Артикул          = СокрЛП(СтрокаТовара["Артикул"]);
   Коэфф             = СтрокаТовара["Коэффициент"];
   БазоваяЕдиница = ?(СтрокаТовара["ФлагБазовойЕдиницы"], Строка(СтрокаТовара["Упаковка"]), "");
   ЕстьСерии         = СтрокаТовара["ЕстьСерии"];
   СтатусУказанияСерий     = СтрокаТовара["СтатусУказанияСерий"];

   мДанные.Установить(  0, ИдТовара ); //Product.Id
   мДанные.Установить(  1, Артикул ); //Product.Marking
   мДанные.Установить(  2, Код ); //Product.Barcode - выгружаем Код позиции номенклатуры, чтобы на
   //терминале была возможность поиска по коду
   мДанные.Установить(  3, ШК ); //Packing.Barcode - штрихкод упаковки
   мДанные.Установить(  4, Наименование ); //Product.Name
   мДанные.Установить(  5, БазоваяЕдиница); //Product.BasePackingId - ид. базовой упаковки (единицы) //или
```

```
    пустая строка, если данная единица не является базовой
    мДанные.Установить(  6, ЕдиницаИзмерения ); //Packing.Id
    мДанные.Установить(  7, ЕдиницаИзмерения ); //Packing.Name - в качестве имени и ид. упаковки
    //используем сокращенное наименование (например, “шт”, “кг”)
    мДанные.Установить(  8, Коэфф ); //Packing.UnitsQuantity
    мДанные.Установить(  9, Характеристика ); //Packing.descr - наименование характеристики или пустая
    //строка, если нет характеристики
    мДанные.Установить( 10, Серия ); //Packing.serial - наименование серии или пустая строка, если нет
    //серии
    мДанные.Установить( 11, Цена ); //Packing.price - цена
    мДанные.Установить( 12, Количество ); //Packing.qty - остаток на складе
    мДанные.Установить( 13, ЕстьСерии ); //Product.withserial - ведется учет по сериям
    мДанные.Установить( 14, СтатусУказанияСерий ); //Product.withsn - есть серийные номера
    мДанные.Установить( 15, ИдТовара ); //Product.КлючСерий - поле для связи с таблицей Серии (если
    //таковая выгружается), ид. позиции номенклатуруры или ид. вида номенклатуры

    //Алко поля
   мДанные.Установить( 16, СтрокаТовара[“Алко”]);
   мДанные.Установить( 17, СтрокаТовара[“АлкоОбъем”]);
   мДанные.Установить( 18, СтрокаТовара[“АлкоКодВ”]);
   мДанные.Установить( 19, СтрокаТовара[“АлкоНаимВ”]);
   мДанные.Установить( 20, СтрокаТовара[“АлкоКрепость”]);
   мДанные.Установить( 21, СтрокаТовара[“Производитель”]);
   мДанные.Установить( 22, СтрокаТовара[“ПроизвИНН”]);
   мДанные.Установить( 23, СтрокаТовара[“ПроизвКПП”]);
   мДанные.Установить( 24, СтрокаТовара[“АлкоКод”]);
   мДанные.Установить( 25, СтрокаТовара["АлкоМарк"]);

    //Добавляем массив данных в выгрузку
    Если Не Соединение.ДобавитьВВыгрузкуТоваров(мДанные) Тогда
      //Произошла ошибка
       ОписаниеОшибки = “”;
       Соединение.ПолучитьОшибку(ОписаниеОшибки);
       Сообщить(ОписаниеОшибки);

       Соединение.СброситьВыгрузкуТоваров(); //Прерываем выгрузку
       Соединение.ОсвободитьРесурсы();
       Возврат;
     КонецЕсли;

  КонецЦикла;

  //Завершение выгрузки
  Если Не Соединение.ЗавершитьВыгрузкуТоваров() Тогда
     ОписаниеОшибки = “”;
     Соединение.ПолучитьОшибку(ОписаниеОшибки);
     Сообщить(ОписаниеОшибки);
  КонецЕсли;

  Соединение.ОсвободитьРесурсы();

Иначе
   //Ошибка при подключении к базе
   ОписаниеОшибки = “”;
   Соединение.ПолучитьОшибку(ОписаниеОшибки);
   Сообщить(ОписаниеОшибки);
КонецЕсли;
```

«1С:Предприятие 7.7»:

```
ПрогИД = "AddIn.Cl.TerminalConnector";
```

```
ПодключитьВнешнююКомпоненту(ПрогИД);
Соединение = СоздатьОбъект(ПрогИД);

Параметры = СоздатьОбъект("СписокЗначений");
Параметры.ДобавитьЗначение(“михаил-пк:10501/3cb8cb43-ca60-42c5-94d9-bb352dcba9e1”); //Строка
// подключения или ид. базы

ИДУстройства = “”;
Если Соединение.Подключить(Параметры, ИДУстройства) <> 0 Тогда
  //Успешное подключение

  ТаблицаНоменклатуры = ПолучитьТаблицуНоменклатурыДляВыгрузки();

  //Описываем схему выгрузки
  мДанные = СоздатьОбъект("СписокЗначений");
  мДанные.ДобавитьЗначение("Product.Id" );
  мДанные.ДобавитьЗначение("Product.Marking" );
  мДанные.ДобавитьЗначение("Product.Barcode" );
  мДанные.ДобавитьЗначение("Packing.Barcode" );
  мДанные.ДобавитьЗначение("Product.Name" );
  мДанные.ДобавитьЗначение("Product.BasePackingId" );
  мДанные.ДобавитьЗначение("Packing.Id" );
  мДанные.ДобавитьЗначение("Packing.Name" );
  мДанные.ДобавитьЗначение("Packing.UnitsQuantity" );
  //Дополнительные поля
  мДанные.ДобавитьЗначение("Packing.descr" );
  мДанные.ДобавитьЗначение("Packing.serial" );
  мДанные.ДобавитьЗначение("Packing.price" );
  мДанные.ДобавитьЗначение("Packing.qty" );
  мДанные.ДобавитьЗначение("Product.withserial" );
  мДанные.ДобавитьЗначение("Product.withsn" );
  мДанные.ДобавитьЗначение("Product.КлючСерий" );
  //Дополнительные поля для работы с алкоголем
  мДанные.ДобавитьЗначение("Product.Алко" );
  мДанные.ДобавитьЗначение("Packing.АлкоОбъем" );
  мДанные.ДобавитьЗначение("Packing.АлкоКодВ" );
  мДанные.ДобавитьЗначение("Packing.АлкоНаимВ" );
  мДанные.ДобавитьЗначение("Product.АлкоКрепость" );
  мДанные.ДобавитьЗначение("Product.Производитель" );
  мДанные.ДобавитьЗначение("Product.ПроизвИНН" );
  мДанные.ДобавитьЗначение("Product.ПроизвКПП" );
  мДанные.ДобавитьЗначение("Packing.АлкоКод" );
  мДанные.ДобавитьЗначение("Packing.АлкоМарк" );

  Соединение.УстановитьПоискПоНаименованиюИАртикулу(1); //Устанавливаем признак, что будет
  //доступен поиск по наименованию товара на терминале

  //Начинаем выгрузку
  Если Соединение.НачатьВыгрузкуТоваров(мДанные) = 0 Тогда
  //Ошибка при начале выгрузки
        ОписаниеОшибки = “”;
        Соединение.ПолучитьОшибку(ОписаниеОшибки);
        Соединение.ОсвободитьРесурсы();

        Сообщить(ОписаниеОшибки);
        Возврат;
  КонецЕсли;

  Пока ТаблицаНоменклатуры.ПолучитьСтроку() = 1 Цикл

     мДанные.УдалитьВсе();
```

```
    Код = СокрЛП(ТаблицаНоменклатуры.Номенклатура.Код); //В качестве идентификатора товара будем выгружать
    код позиции номенклатуры
    Наименование              = СокрЛП(ТаблицаНоменклатуры.Номенклатура.Наименование);
    ШК  = Запрос.Штрихкод;
    ЕдиницаИзмерения          = СокрЛП(ТаблицаНоменклатуры.ЕдиницаИзмерения);
    Характеристика           = СокрЛП(ТаблицаНоменклатуры.Характеристика);
    Серия             = СокрЛП(ТаблицаНоменклатуры.Серия);
    Цена              = ТаблицаНоменклатуры.Цена;
    Количество        = ТаблицаНоменклатуры.Количество;
    Артикул          = СокрЛП(ТаблицаНоменклатуры.Артикул);
    ….

    мДанные.ДобавитьЗначение(Код);
    мДанные.ДобавитьЗначение(Артикул);
    мДанные.ДобавитьЗначение(Код);
    мДанные.ДобавитьЗначение(ШК);
    мДанные.ДобавитьЗначение(Наименование);
    ….
    мДанные.ДобавитьЗначение(СокрЛП(ТаблицаНоменклатуры.АлкоМарк));

    //Добавляем массив данных в выгрузку
    Если Соединение.ДобавитьВВыгрузкуТоваров(мДанные) = 0 Тогда
      //Произошла ошибка
       ОписаниеОшибки = “”;
       Соединение.ПолучитьОшибку(ОписаниеОшибки);
       Сообщить(ОписаниеОшибки);

       Соединение.СброситьВыгрузкуТоваров(); //Прерываем выгрузку
       Соединение.ОсвободитьРесурсы();
       Возврат;
     КонецЕсли;

  КонецЦикла;

  //Завершение выгрузки
  Если Соединение.ЗавершитьВыгрузкуТоваров() = 0 Тогда
     ОписаниеОшибки = “”;
     Соединение.ПолучитьОшибку(ОписаниеОшибки);
     Сообщить(ОписаниеОшибки);
  КонецЕсли;

  Соединение.ОсвободитьРесурсы();


Иначе
  //Ошибка при подключении к базе
  ОписаниеОшибки = “”;
  Соединение.ПолучитьОшибку(ОписаниеОшибки);
КонецЕсли;
```