Знакомство с RFID

Последние изменения: 2024-03-26

RFID (Radio Frequency Identification – «радиочастотная идентификация») – это набор разнородных технологий и стандартов, обеспечивающих бесконтактную идентификацию чего-либо по радиоканалу на расстояниях от нескольких сантиметров до сотен метров. Происходит такая идентификация путем обмена данными между специальным устройством чтения, называемым RFID-считыватель, и специальными метками с антенной и микрочипом, которые наносятся на интересующие объекты.

При этом понятие «RFID-считыватель» (RFID-ридер), микрочипы и радиодиапазон используемых частот – единственное, что объединяет между собой все те технологии, стандарты и оборудование, которые могут скрываться за аббревиатурой RFID.

Считыватели и метки

Любая RFID-система состоит из RFID-считывателя (стационарного или мобильного) и RFID-меток (чаще всего в виде тонких этикеток или толстых корпусных меток).

Все без исключения RFID-считыватели одновременно являются и «RFID-писателями», т.к. и чтение, и запись, происходят путем отправки RFID-считывателем определенных команд по радиопротоколу. Эти команды принимаются и исполняются микрочипами меток. Поэтому RFID-принтеры, используемые для печати на рулоны RFID-этикеток, можно отнести к RFID-считывателям.

Важной частью RFID-считывателя являются антенны (или одна антенна), которые во многом определяют дальность и качество считывания. От формы и материала антенны зависит распределение энергии излучаемой волны в пространстве. Основной характеристикой антенны является её направленность (по-английски «gain») – число, условно показывающее количество энергии, излучаемой антенной в определенном направлении. Без потери физического смысла можно сказать, что если gain = 1, то излучение равномерно распределено во все стороны, а если gain > 19, то большая часть энергии направляется куда-то сконцентрированным лучом.

Обзор частот RFID

На текущий момент «Wonderfid Link» ориентирована только на частоты UHF (Ultra High-Frequency, сверхвысокие частоты) в применении к меткам стандарта Class 1 Gen 2 (стандарт ISO 18000-6C).

На практике все используемые в настоящее время технологии RFID условно подразделяются на следующие группы:

Диапазон
Расшифровка
Особенности
Особенности применения в RFID
LF
Low-Frequency (низкие частоты)
Технологии и оборудование для работы на частотах условно 125-134 КГц. Глубокое проникновение электромагнитных волн на таких частотах в воду (8 метров), живую ткань (2 м) и металлы (~2 мм) обеспечивает их применение в такой области как маркировка скота, домашних и прочих животных. А возможность ограничения зоны считывания несколькими сантиметрами позволяет применять их в системах контроля доступа (но не всегда подходит для смарткарт, т.к. передача данных на такой частоте слишком медленная).

HF
High-Frequency (высокие частоты)
Технологии и оборудование для работы на частотах условно 5-7 МГц, 13,56 МГц. Возможность ограничения зоны считывания несколькими сантиметрами позволяет применять их в системах контроля доступа и оплаты – пропуска, карточки, ключи домофона, сотовые телефоны и т.п.
1. HF лучше, чем UHF, проникает в воду, металлы, живую ткань. 2. Каждая метка HF снабжена уникальным неизменным кодом метки, прошитым на стадии производства. 3. Дальность считывания у HF невысока. 4. Скорость считывания для меток HF не может составлять сотни меток в секунду.
UHF
Ultra High-Frequency (сверхвысокие частоты)
Технологии и оборудование для работы на частотах 433 МГц, 860-960 МГц, 2,4-2,45 ГГц и 5,2-5,8 ГГц.
1. UHF позволяет передавать в секунду больше данных – значит, скорость считывания меток выше. 2. UHF читает метки с бóльшего расстояния, чем HF или LF. 3. Для меток UHF нет необходимости придумывать свои схемы кодирования, изобретать таблицы поиска соответсвтия и т.п. – существуют стандарты GS1 Tag Data Standard и ISO/IEC, в которых всё уже придумано. 4.Для UHF не существует никакого «кода RFID метки» и тем более «уникального кода RFID метки». Строка «3024000003320C4063A23312» (TAG ID), которая читается демопрограммой – это не уникальный код и не просто строка, а шестнадцатеричная запись бинарно закодированного электронного кода объекта (EPC или UII), на который как бы нанесена купленная вами метка (именно поэтому там есть буквы от A до F, но вы никогда не встретите в TAG ID буквы от G до Z). Метка идентифицирует не себя, а объект, на который её клеят. И перед началом использования метки от вас ожидают, что вы самостоятельно пропишите в метку свои EPC или UII (т.е. код товара/объекта/контейнера/документа, а также (возможно) его серийный номер), после чего метка при чтении будет возвращать именно их (см. ниже «Понятие электронного кода объекта»). 5. Для UHF существует понятие «номер чипа», но, в отличие от HF, номер чипа может отсутствовать либо быть неуникальным. Недостаток в том, что этот номер расположен в отдельном банке памяти и медленно читается.

На практике наиболее важным параметром является скорость считывания меток, которая выше у меток на UHF частотах. Кроме того, частоты UHF позволяют использовать более короткие антенны. Поэтому, несмотря на то, что человеческое тело, фольга, бочки с жидкостью и т.д. представляют для волн UHF непреодолимое препятствие, именно UHF и комбинированные UHF/HF RFID технологии развиваются наиболее активно.

От используемой частоты напрямую зависят скорость и расстояние передачи данных, а также габариты антенн (как у считывателей, так и у меток). Чем ниже частота (больше длина волны), тем длиннее должны быть антенны. Соответственно, чем выше частота (UHF), тем меньше может быть метка.

При этом, чем выше скорость передачи данных, тем быстрее должны работать микрочипы считывателей и меток. Именно скорость чипов долгое время сдерживала развитие технологий на частотах UHF.

Для работы чипа необходимо электропитание. Электропитание может поступить либо из батарейки (тогда это называется «активная метка») либо из энергии волн, которыми RFID-считыватель облучает метки (и тогда это «пассивная метка»). Пассивные метки наиболее распространены, т.к. дешевле в производстве и использовании. Опять же, чем выше частота, тем быстрее чип пассивной метки заряжается от энергии волн.

Технологии RFID на разных частотах отличаются не только физическими характеристиками, но и тем, как устроены считыватели и метки, какие возможны операции и т.п. Например, важным отличием меток HF от меток UHF является то, что каждая метка HF имеет свой совершенно уникальный код, прошиваемый еще на стадии производства. А стандарт на метки UHF допускает метки как вообще без уникального номера чипа, так и с коротким номером (например, в 32 бита), уникальность которого гарантируется только в пределах пары лет при массовом выпуске меток, причем этот номер читается медленно и не подходит для массовой инвентаризации.

Не нашли что искали?

Задать вопрос в техническую поддержку

Маркировка объектов при помощи RFID

Последние изменения: 2024-03-26

Понятие электронного кода объекта

Электронный код объекта – это цепочка байтов, которую можно декодировать по заранее определенным правилам, чтобы получить на выходе данные об объекте, маркированном RFID-меткой. Т.е. объект идентифицирует не сама метка, и не номер метки, а электронный код объекта, прописанный в ней.

Для разных типов объектов (автомобили, грузы, товары, книги и т.п.) стандартами определен свой набор данных, который считается достаточным для идентификации соответствующего объекта. Предполагается, что перед нанесением RFID-метки на объект, в неё будет прописан закодированный электронный код этого объекта.

В настоящий момент существуют две системы стандартов электронных кодов объектов для использования в RFID. Первые разрабатывает организация EPCglobal GS1. Вторые разрабатывает ISO/IEC. Обе системы частично пересекаются в вопросах, что маркировать и как кодировать. При чтении меток программа всегда имеет возможность понять, по какому стандарту закодирована метка. Вопрос, какой стандарт следует использовать для кодирования, решается отдельно в каждой конкретной области применения, для каждого конкретного типа маркируемых объектов.

Метки HF кодируются по стандартам ISO/IEC. Метки UHF кодируются как по стандартам ISO/IEC (реже), так и по стандарту EPCglobal GS1 (чаще).

Архитекторы UHF RFID на метках Class 1 Gen 2 выстроили довольно сложную систему идентификации объектов, которая базируется на существующих стандартах международных организаций GS1 и ISO/IEC по идентификации товаров, грузовых контейнеров, автомобилей, книг, авиабагажа и т.п. Результирующие решения, реализованные в «железе» Class 1 Gen 2, кардинально отличаются от того, что ожидает от RFID-учета любой неподготовленный заранее «технарь».

В первую очередь это касается вопроса об «уникальных номерах меток».

Для HF каждая без исключения метка имеет уникальный неизменяемый номер метки, прошитый на стадии производства. Этот номер уникально идентифицирует саму метку. В памяти, доступной для записи, метка несет информацию о маркируемом объекте.

Таким образом, систему на HF можно построить на базе простого соответствия уникального кода метки объектам базы данных учетной системы, а в память метки, отвечающую за данные о маркируемом объекте, вообще ничего не писать.

Для UHF Class 1 Gen 2 метка может не иметь уникального номера. Даже если номер предусмотрен производителем, он необязательно уникальный. И даже если он уникальный, этот код расположен в отдельном специальном банке памяти метки, который медленно читается и не подходит для массовой инвентаризации (см. подробности ниже).

Таким образом, строить систему на UHF путем соответствия заранее кем-то назначенных «уникальных кодов» объектам своей базы данных не всегда возможно (либо возможно, но не всегда практично с точки зрения скорости чтения). «Уникальные коды» зачастую придется генерировать и прописывать самим.

На заре UHF RFID предполагалось, что RFID-метки будут служить простой альтернативой GS1 штрихкодам (например, EAN13, которые одинаковы для всех идентичных экземпляров товара), и что прошиваться и наноситься на товары они будут еще на этапе производства. Просто вместо того, чтобы заказывать обычные бирки или пачки с заранее напечатанным EAN13, производитель будет заказывать «умные» бирки со штрихкодом + встроенными чипами с заранее прописанным аналогом EAN13, совершенно одинаковым для экземпляров товара. Учета уникальных единиц товара не предполагалось.

В системе с неуникальными метками человек должен был подходить с товаром на кассу и система пробивала бы его – какой-то товар просто «по штрихкоду», какой-то «по штрихкоду из RFID-метки» (метки Class 0). В итоге даже в Class 1 Gen 2 есть возможность подсчета точного количества меток на кассе, даже если все метки идентичны!

Недостатком такой системы служат непреодолимые сложности при инвентаризации. Хотя на кассе система точно подсчитывает количество товара, этого невозможно сделать в мобильном режиме в торговом зале, т.к. человек со считывателем может несколько раз пройти мимо одной и той же полки с разрывом в несколько минут, и система несколько раз «пробъет» товар на ней в результаты инвентаризации.

Решением проблемы является наличие для каждого экземпляра товара некоего уникального кода, в дополнение к коду товара, что и было реализовано в Class 1 Gen 2. Для маркировки товаров по схеме GS1 EPC таким кодом был выбран серийный номер – числовой для меток в 128 бит и строковый для меток от 512 бит и более (имеется в виду размер банка EPC/UII). Для учета библиотечных фондов, основных средств, сотрудников и т.п. были придуманы еще более сложные академические схемы, основанные на системе Relative OIDs и классификации ASN.1

Ответственность за назначение серийных и прочих номеров была возложена на пользователя меток. Серийные номера могут быть «фиктивными», т.е. никак не отражаться в учете и назначаться просто по счетчику, а могут быть реальными. В идеале от организации требовалось организовать у себя систему учета на уровне отдельных объектов, по серийным номерам.

Во всех случаях предполагалось, что код объекта в метку прошьет пользователь, а не производитель. С появлением новых чипов и осознанием производителями того факта, что пользователи ожидают от них заранее прошитых уникальных меток, ситуация постепенно меняется, но не кардинально.

«Кошмар 2050 года»

Если представить, что на дворе 2050 год и всё вокруг промаркировано RFID-метками, то ситуация будет кардинально отличается от той, когда всё вокруг промаркировано штрихкодами. Потому что штрихкод считывается только тот, на который был направлен сканер. А RFID-метки читаются все вокруг, сразу несколько, и с учетом переотражения сигнала потенциально могут считаться любые метки в радиусе нескольких метров.

Используя штрихкоды, почти всегда можно было закрыть глаза на все стандарты, ведь всегда известно, какой штрихкод наш, где он наклеен и как выглядит. Какие-то стандарты могут потребоваться только при маркировке для продаж в крупных супермаркетах, при маркировке грузов в международной логистике и прочих крупных затеях. Это разрешено, т.к. сканер штрихкодов читает только то, на что мы его направили. Невозможно себе представить, что мы направили луч на один штрихкод, а считался совсем другой с обратной стороны коробки. Внезапное чтение «левого» штрихкода с пачки сигарет, которая лежит в кармане кладовщика, также невозможно.

С RFID всё иначе. Каждая промаркированная RFID пачка сигарет, маркированная личная одежда персонала, каждый документ (паспорта, права) – любая мелочь в охвате нескольких метров может быть прочитана вашим считывателем как своя.

Поэтому станут актуальными подобные запреты:

Имеется в виду: «не проносить с собой чужих RFID-меток!». К сожалению, такой запрет не подходит для магазина.

Итак, RFID читает всё вокруг. Для проверки прихода и отгрузки можно использовать RFID-тоннель, который читает только то, что проходит сквозь него. Но, опять же, RFID-тоннель не подходит для задачи быстрой инвентаризации (т.к. перетаскать весь товар к тоннелю и обратно – это долго).

Таким образом, внедряя RFID-систему, придется использовать международные стандарты, специализированное оборудование.

RFID Class 1 Generation 2 (UHF)

Стандарт Class 1 Generation 2 (Class 1 Gen 2 или просто Gen2, второе поколение первого класса) – это набор документов, разработанных коммерческой организацией «EPCglobal, Inc.», в которых подробно описано своего рода «техническое задание» на чипы RFID-меток и работу RFID-считывателей (стандарт ISO18000-6C).

Основное в стандарте Class 1 Generation 2:

1. Чипы меток и считыватели должны работать на частотах UHF 860-960 МГц
   (при этом и считывателям, и чипам меток теоретически не запрещается в дополнение к UHF поддерживать и любые другие частоты).
2. Чип каждой метки может иметь свой идентификационный номер, прошитый производителем еще на стадии производства
   (при этом в стандарте прописана структура номера: он должен начинаться на «E0», «E2» или «E3», и опционально содержать номер производителя, номер модели и серийный номер чипа. EPCglobal занимается регистрацией производителей чипов и выдает им те самые номера производителя).
3. Чипы меток должны поддерживать не только чтение, но и запись данных.
4. В чипе должен присутствовать специальный банк памяти для хранения уникального идентификатора маркируемого объекта (так называемого EPC/UII), прошиваемый на этапе начала пользования меткой
   (этот банк записывается пользователем метки. т.е. это не уникальный код, не номер чипа, записываемый производителем, и вообще не имеет к номеру чипа никакого отношения).
5. Чипы в метках должны позволять задавать пароль доступа на чтение или запись данных.
6. Чипы в метках должны позволять «прожигать» данные намертво, так чтобы их уже нельзя было переписать.
7. Чипы в метках должны позволять безвозвратно стирать с них информацию, производить так называемое «убийство» метки
   (в данном случае самоубийство).
8. Чипы в метках должны позволять задавать пароль на эту функцию «убийства», в дополнение к паролю на доступ к чтению/записи.

На уровне радио-протокола обмена между чипами меток Class 1 Gen 2 и считывателем UHF можно производить следующие операции:

1. Операция отбора меток по определенным условиям.
2. Операция инвентаризации отобранных меток (самая быстрая и надежная).
3. Операция чтения содержимого конкретных банков чипа RFID-метки.
4. Операция записи какого-то заранее известного значения (константы) в определенные места конкретных банков чипа RFID-метки (пишет все чипы, которые подошли под ранее заданный критерий).
5. Операция «прожига намертво» содержимого конкретного банка чипа RFID-метки.
6. Операция блокирования/разблокирования банков.
7. Операция «убийства» чипа RFID-метки.

Метки стандарта Class 1 Generation 2

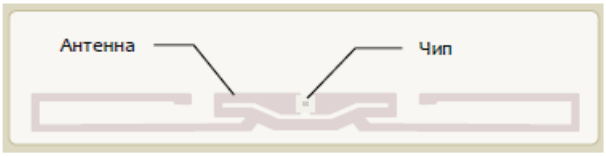
Стандарт на Class 1 Generation 2 описывает только частоты, протоколы обмена и некоторые алгоритмы работы (или советы по алгоритмам) для чипов, используемых в RFID-метках. Помимо чипа, метка состоит из антенны и субстрата для крепления метки на объект. В стандарте ничего не сказано о креплении на металл, надежности приклеивания или размерах метки. Всё, что нужно для надежного крепления и хорошего считывания, –

отдается на откуп производителя конкретных меток.

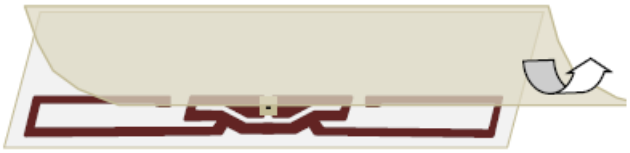
Метки могут быть любыми: тонкие самоклеющиеся бумажные и синтетические в виде этикетки, толстые пластмассовые корпусные, стеклянные вживляемые, съедобные и т.д.

RFID-метка в виде этикетки, вид спереди

RFID-метка в виде этикетки, вид на просвет



RFID-метка в виде этикетки, вид сзади (подложка отклеена)



RFID-метка в виде этикетки, наклеена на объект

Чип в метке стандарта Class 1 Gen 2 позволяет читать из 4 банков памяти разного размера и писать в 3 из них. Тот единственный банк, из которого можно только читать (но нельзя писать), – это банк №2, в котором должен храниться уникальный номер чипа, присваиваемый еще на стадии производства чипа.

Структура данных в чипе стандарта Gen2 (4 банка памяти):

Банк №0	(RESERVED)	Пароль уничтожения (4 байта)	Пароль для чтения (4 байта)	+что-нибудь еще...
Банк №1	(EPC)	Заголовок (4 байта)	EPC/UII* (от 12 байт)	XPC + еще...
Банк №2	(TID)	Номер чипа (от 4х до 12ти байт)		
Банк №3	(USER)	Пользовательские данные. Зависит от чипа метки: банк может отсутствовать, а может быть в несколько килобайт		

* EPC - Electronic Product Code (Электронный код продукта), UII – Unique Item Identifier (Уникальный идентификатор объекта), см. специальный раздел ниже.

Пяти и больше банков в метке Class 1 Gen 2 быть не может, т.к. в протоколе общения считывателя с меткой номер банка кодируется всего двумя битами (итого получаются сочетания «00», «01», «10» и «11»).

Принципы идентификации объектов с помощью меток Class 1 Generation 2

Архитекторы UHF RFID разрабатывали всю систему исходя из следующего принципа идентификации:

1. Перед маркировкой объекта в метку записывается уникальный номер идентифицируемого объекта (в виде EPC или UI). Метка крепится к объекту. Таким образом, EPC/UII в метке идентифицирует объект, к которому прикреплена метка. Метки инвентаризуются по EPC/UII, со скоростью примерно тысяча меток в секунду.

2. В метке дополнительно может храниться номер чипа, который идентифицирует саму метку, а не тот объект, к которому она прикреплена. Номер чипа может отсутствовать и не обязан быть уникальным. Номер чипа никак не связан с EPC/UII и не имеет никакого отношения к идентифицируемому объекту.

Правильное понимание того, как устроен «уникальный код» RFID метки Class 1 Gen 2 (UHF RFID):

1. «Уникальный код» метки (TAG ID) сам по себе не уникален. Большинство производителей поставят вам метки (рулон или коробку) с совершенно идентичными кодами. Ожидается, что вы сами будете следить за

уникальностью, соблюдая международные стандарты. Вы пропишете код товара/контейнера/документа и, возможно, его уникальный серийный номер в код RFID-метки, после чего метки при инвентаризации будут видны как разные.

2. «Уникальный код» метки (TAG ID) представляет собой электронный код продукта. В него следует закодировать номер товара/объекта/контейнера/документа согласно вашей базе, а также серийным номером помечаемого объекта (опционально). Почти на все случаи жизни уже придуманы и приняты какие-то стандарты кодирования. Придумывать какие-то свои схемы кодирования и записывать их в код метки не запрещается, но и не приветствуется.
3. Ожидается, что уникальность обеспечивает система идентификации в вашей организации, а не производитель меток. Ожидается, что вы сами пропишете код товара/объекта или контейнера/документа и его уникальный серийный номер в код RFID-метки перед наклейкой её на интересующий объект. Прошитая метка будет нести в себе ваш код и серийный номер, будет возвращать их при инвентаризации.
   Этот пункт на практике очень сложно реализуем, особенно в распределенных системах учета. Требуется придумывать какие-то диапазоны номеров, реплицировать данные и т.п. Одним из возможных решений является Chip Based Serialization.

4. Прочитанный код любой метки, например «3024000003320C4063A23312», следует декодировать и вытащить из него информацию о маркированном объекте.
5. В Wonderfid Link уже реализовано большинство стандартных методов кодирования/декодирования, и уже есть возможность работать с метками в терминах кодов товаров, номеров книг, серийных номеров, штрихкодов EAN13 и т.п.
6. Если «уникальный код» метки (TAG ID) записали за вас – значит скорее всего это никакой не уникальный код, и ценность его минимальна.

Вопрос-ответ

Но продавец говорит, что каждая его метка имеет уникальный код!

Продавец в данном случае говорит об уникальном номере у каждого RFID-чипа, используемого им при производстве меток. У любого чипа RFID-метки Class 1 Gen 2 (UHF RFID) согласно стандарту ISO18000-6C может быть код (необязательно уникальный). У любой такой метки в любой момент можно узнать номер используемого в ней чипа, но это значительно более медленная и ненадежная операция, чем инвентаризация RFID, и полагаться на неё в инвентаризации нельзя. Инвентаризация по кодам товаров позволяет читать до 1000 меток/сек. Инвентаризация по номерам чипов едва ли прочтет 5 меток/сек и почти никогда не прочтет все метки, если их лежит больше 1 шт, и не подходит для товарной инвентаризации. Для целей товарной инвентаризации исключительно всегда следует использовать банк EPC/UII, в котором метка хранит код объекта, на который она нанесена, и который возвращается считывателю при инвентаризации.

Наше ТЗ содержит простую таблицу соответствия кодов меток объектам нашей базы данных!

Ваше ТЗ ересь (в прямом смысле), т.к. идет против международных стандартов. Код метки не является случайным уникальным числом, а представляет собой «карточку», которую следует заполнить данными из вашей базы. В зависимости от характера маркируемых вами объектов (товары это, документы, книги или другое имущество) заранее предусмотрены стандартные схемы заполнения этой «карточки». Другие банки меток (в частности, пользовательский банк) не предусмотрены для быстрой инвентаризации, их используют для других задач (хранение расширенной информации для операций с отдельными метками, выявление клонов меток и подделок). Для целей инвентаризации исключительно всегда следует использовать банк EPC/UII, в котором метка хранит данные объекта, на который она нанесена, заполненные по правилам международных стандартов. Эти данные возвращается считывателю при инвентаризации. Изобретение своих схем кодирования не запрещается, но и не приветствуется.

Но наше ТЗ содержит простую таблицу соответствия кодов меток объектам нашей базы данных!

В этом случае убедитесь, что вы предварительно сами прописываете банк EPC/UII, и используете при этом стандартную действительно глобально уникальную схему кодирования.

EPC и UII как электронные коды объектов

Банк №1 для RFID-меток Class 1 Gen 2 (который также может называться «EPC-банк», «банк EPC», «банк UII»,

«второй банк» или «банк 01») содержит в себе электронный код объекта, представленный либо в виде EPC, либо в виде UII (третьего варианта нет, т.к. за выбор отвечает один единственный бит в заголовке банка). За стандартизацию кодирования EPC отвечает международная организация GS1, за стандартизацию UII отвечают организации ISO/IEC.

EPC (Electronic Product Code) – это способ идентификации конкретных единиц товаров, мест хранения, документов и т.п., который используется при маркировке объектов RFID-метками Class 1 Gen 2 по стандарту EPCglobal GS1.

В RFID-метку EPC записывается при помощи нулей и единиц. Перевод EPC в нули и единицы называется бинарным кодированием EPC, и уже реализовано в Wonderfid Link. Из метки EPC считывается точно так же в виде нулей и единиц (обычно в виде шестнадцатеричного представления закодированных байтов, например «3024000003320C4063A23312»), и чтобы получить из них код компании или серийный номер товара, необходимо произвести декодирование.

UII (Unique Item Identifier) – это способ идентификации конкретных единиц имущества, библиотечных элементов, грузов, бейджей сотрудников, документов и т.д., который используется при маркировке объектов любыми RFID-метками по стандартам ISO/IEC. Существует целый ряд стандартов ISO/IEC, расчитанный каждый на свою область применения, из которых основным является ISO-15961.

В RFID-метку UII записывается также при помощи нулей и единиц. Перевод UII в ноли и единицы называется бинарным кодированием UII, и уже реализовано в Wonderfid Link. Из метки UII считывается точно так же в виде нулей и единиц (обычно в виде шестнадцатеричного представления закодированных байтов, например «069100051CBE991A14»), и чтобы получить из них данные маркируемого объекта (например, номер книги), необходимо произвести декодирование.

Соответственно, для маркировки тех или иных объектов уже придуманы соответствующие схемы кодирования банка EPC/UII либо как EPC, либо как UII:

- Для маркировки товаров (от джинс до ювелирки) – схема SGTIN для EPC по стандарту GS1.
- Для маркировки оборудования – либо схема GIAI для EPC по стандарту GS1, либо схема di по стандарту ISO.
- Для маркировки шин – схема di по стандарту ISO.
- Для маркировки в библиотеках – стандарт ISO 28560.
- Для маркировки автомобилей – схема di по стандарту ISO.
- Для маркировки зданий и комнат – схема SGLN для EPC по стандарту GS1.
- Для маркировки пробирок, поддонов – схема GIAI для EPC по стандарту GS1.

Не нашли что искали?

Задать вопрос в техническую поддержку

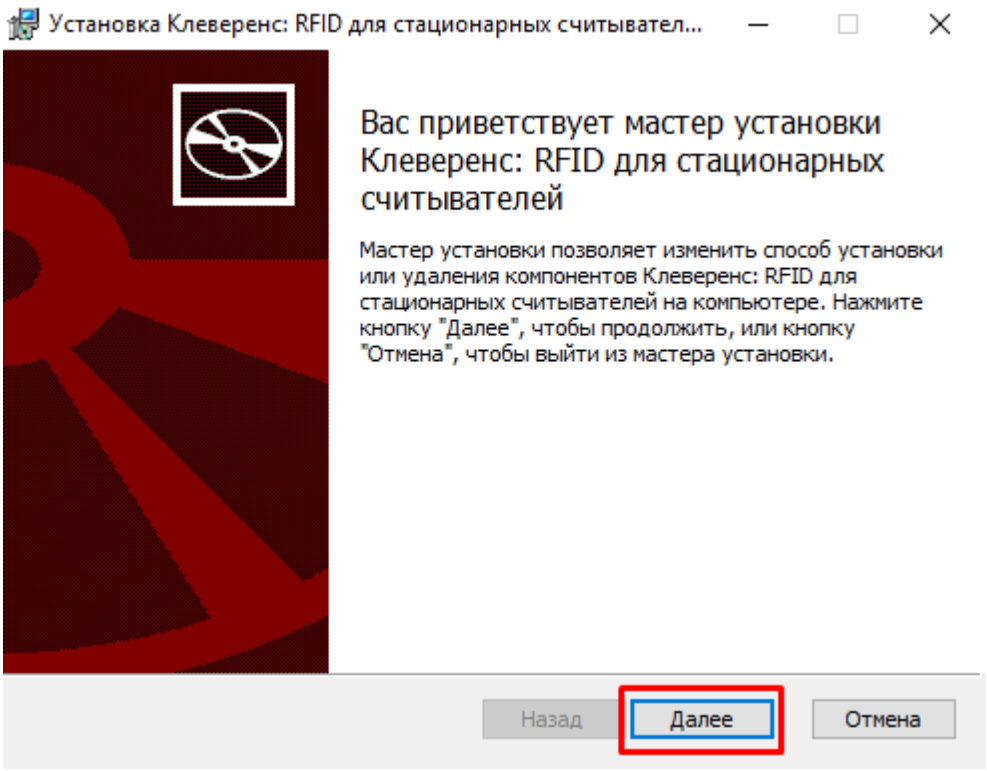
Установка и настройка «Wonderfid™ Link» на ПК

Последние изменения: 2024-03-26

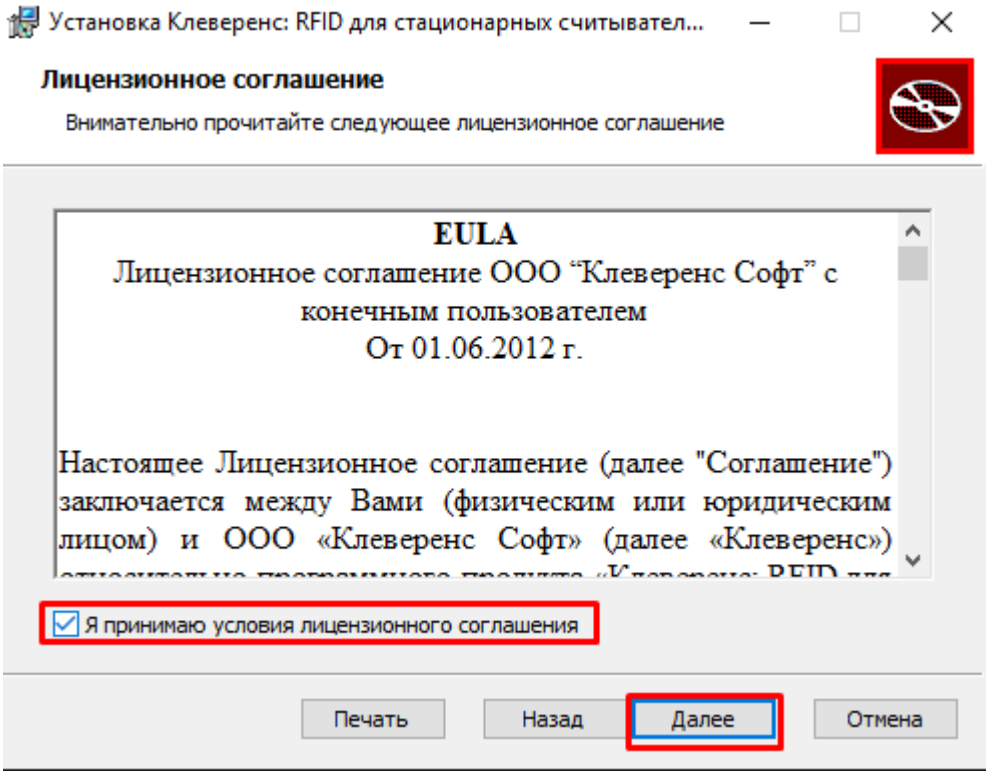
Установка «Wonderfid™ Link»

Для упрощения установки «Wonderfid™ Link», необходимо скачать специальный мастер установки, который поможет установить все компоненты (программы), необходимые для работы.

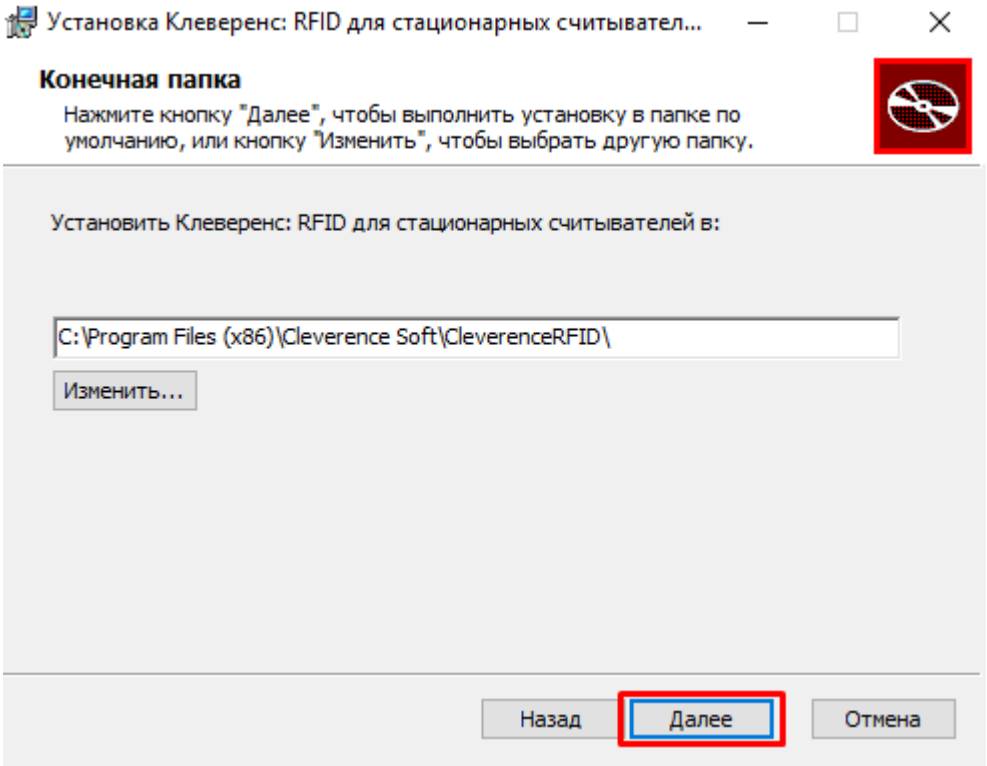
После запуска скачанного файла, откроется окно установщика.



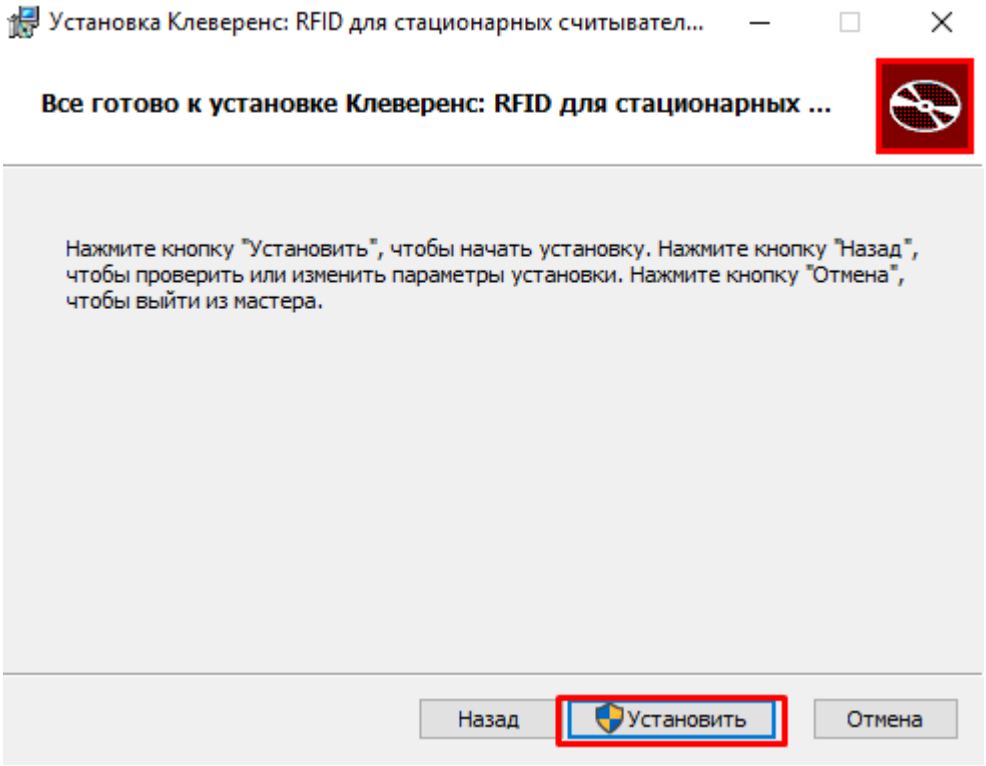
Примите условия лицензионного соглашения.



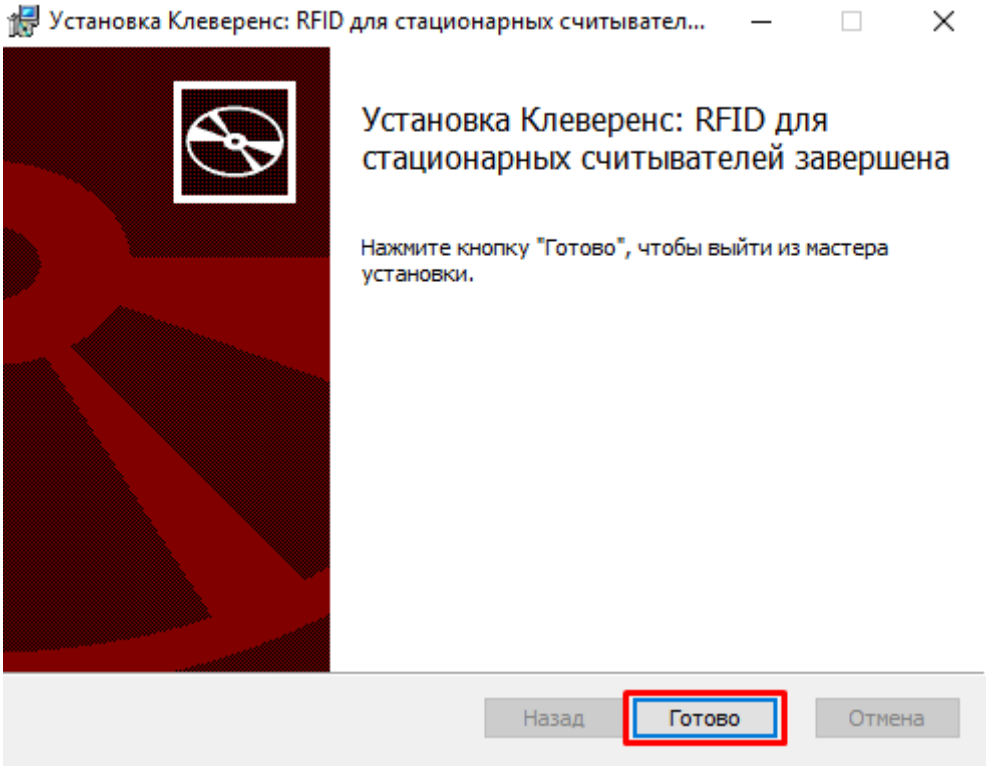
Подтвердите установку продукта в папку по умолчанию или выберите другую с помощью кнопки «Изменить».



Для запуска установки продукта нажмите кнопку «Установить».

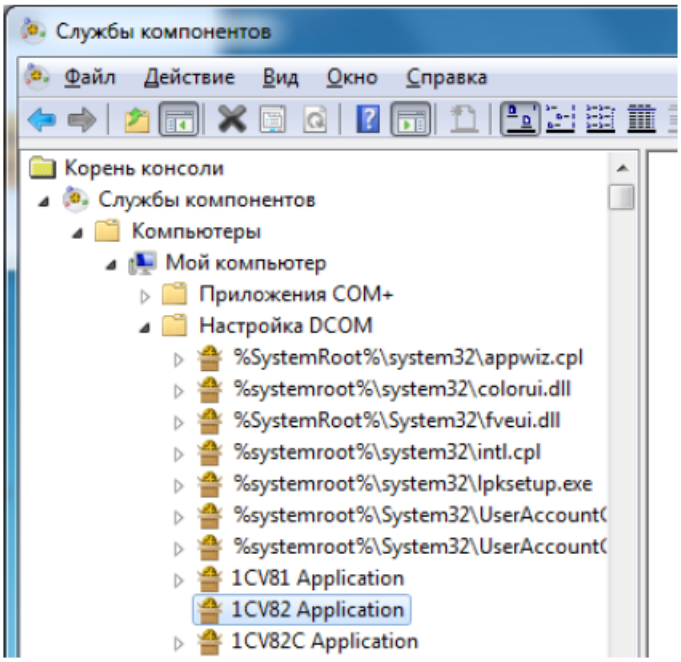


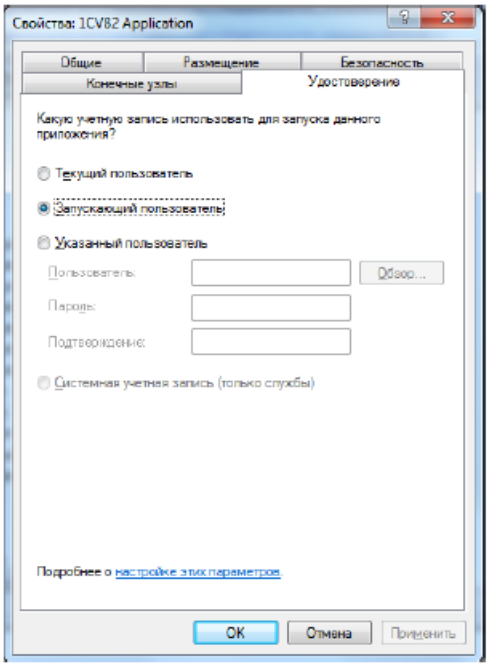
После завершения установки нажмите кнопку «Готово».



Дополнительная настройка «Wonderfid™ Link»

На операционных системах версий «Home» (Windows 7 Home, Windows 7 Home Premium, Windows XP Home и т.п.) настройки политики безопасности по умолчанию не позволяют продукту посылать события в «1С:Предприятие». Из-за этого в версиях «Home» может не работать асинхронное считывание RFID-меток. Попробовать решить эту проблему можно следующим образом:





На некоторых компьютерах узлы «1CV81 Application», «1CV82 Application» и «1CV82C Application» по неизвестной причине могут отсутствовать – в этом случае попробуйте переустановить 1С, либо используйте компоненту на другом компьютере.

Не нашли что искали?

Задать вопрос в техническую поддержку

Установка и настройка RFID считывателей

Последние изменения: 2024-03-26

> Если у вас нет под рукой RFID-считывателя — не беда. Компонента позволяет работать в «виртуальном режиме», имитируя считывания и записи несуществующих меток несуществующими считывателями. Пользуясь «виртуальным режимом» можно написать и отладить логику программы без необходимости иметь дело с реальным оборудованием.

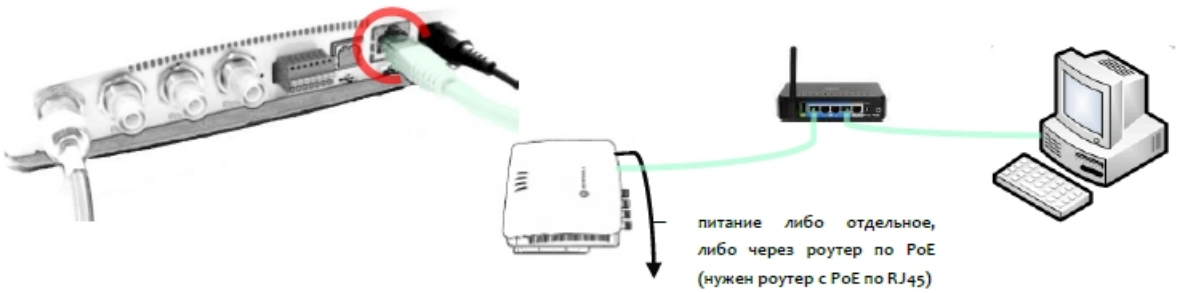
Перед началом работы с RFID считывателем, необходимо произвести его предварительную настройку. В зависимости от модели настройки будут отличаться.

Установка и настройка Motorola FX7400

Считыватель Motorola FX7400 способен работать по сети в двух конфигурациях:

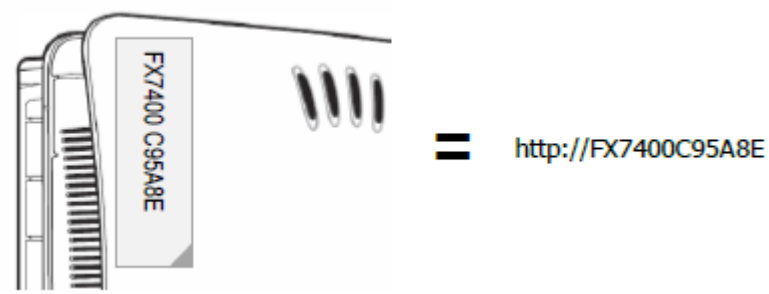
1. через разъем Ethernet (RJ45), путем подключения через роутер либо напрямую к другому ПК;
2. через разъем mini-USB типа A, путем подключения к ПК через драйвер виртуальной сети RNDIS.

Подключение Motorola FX7400 через витую пару (кабель Ethernet, разъем RJ45)



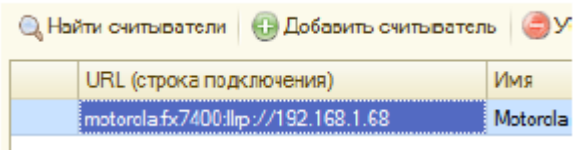
В этой конфигурации считыватель доступен по сети:

- по IP, который ему должен выдать сам роутер или DHCP сервер
- по сетевому имени с наклейки на крышке считывателя (для сетей с DHCP):



> Не зная IP или сетевого имени невозможно будет подключиться к считывателю и настроить его!

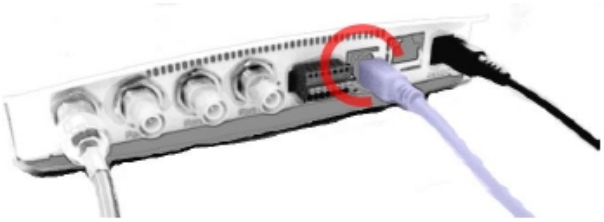
Узнать IP считывателя можно путем поиска считывателей в обработке CleverenceRFID:

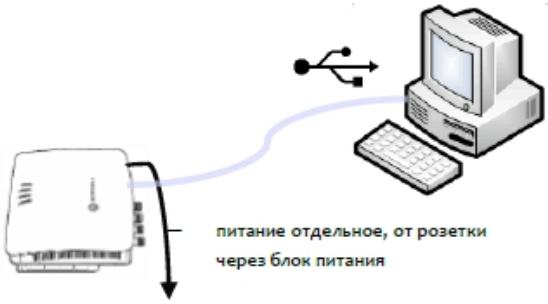


Подключение Motorola FX7400 через кабель USB

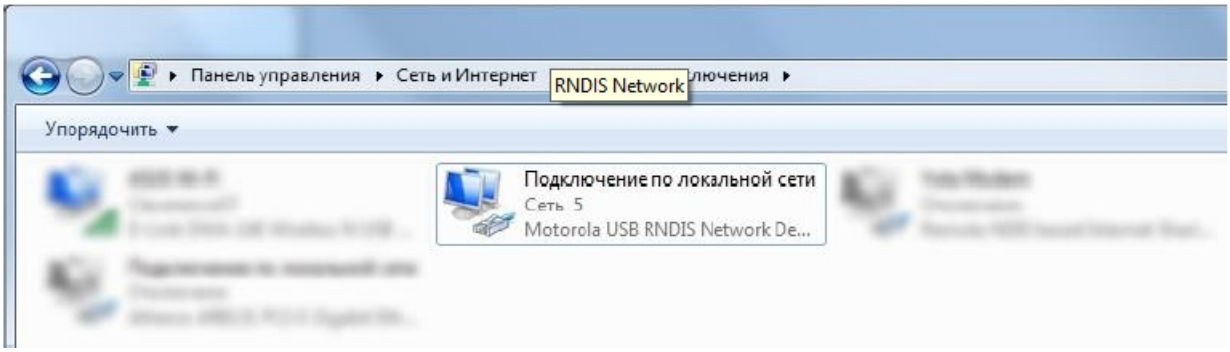
Для подключения потребуется кабель с разъемами USB-A («прямоугольник») на mini-USB-A («квадратик»).

> Перед подключением кабеля USB следует скачать и установить драйвер виртуальной сети с сайта Моторола.



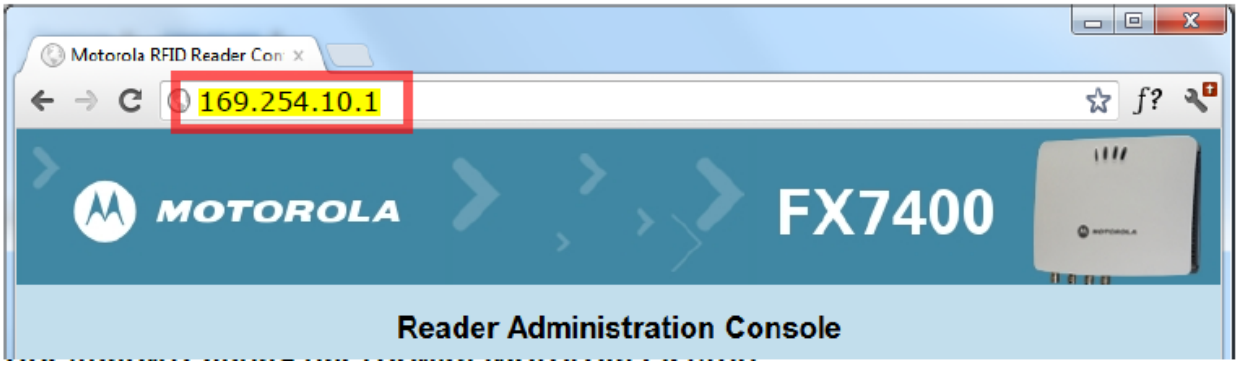


После установки драйвера виртуальной сети и подключения кабеля USB в системе должно появиться новое сетевое подключение с адаптером «Motorola USB RNDIS Network Device»:



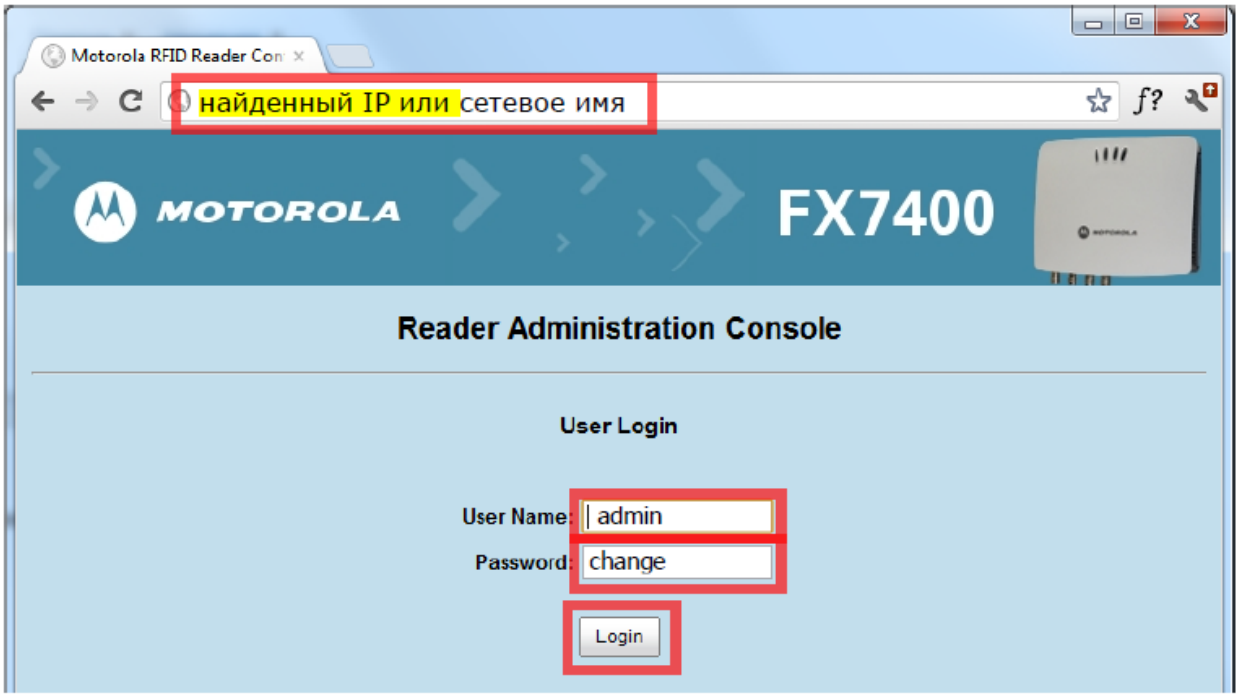
При отключении кабеля USB соединение исчезает из списка. При подключении появляется снова (если не появилось — нажмите F5, чтобы обновить список).

В новой виртуальной сети IP считывателя всегда будет равен «169.254.10.1», его следует ввести в адресной строке вашего браузера:

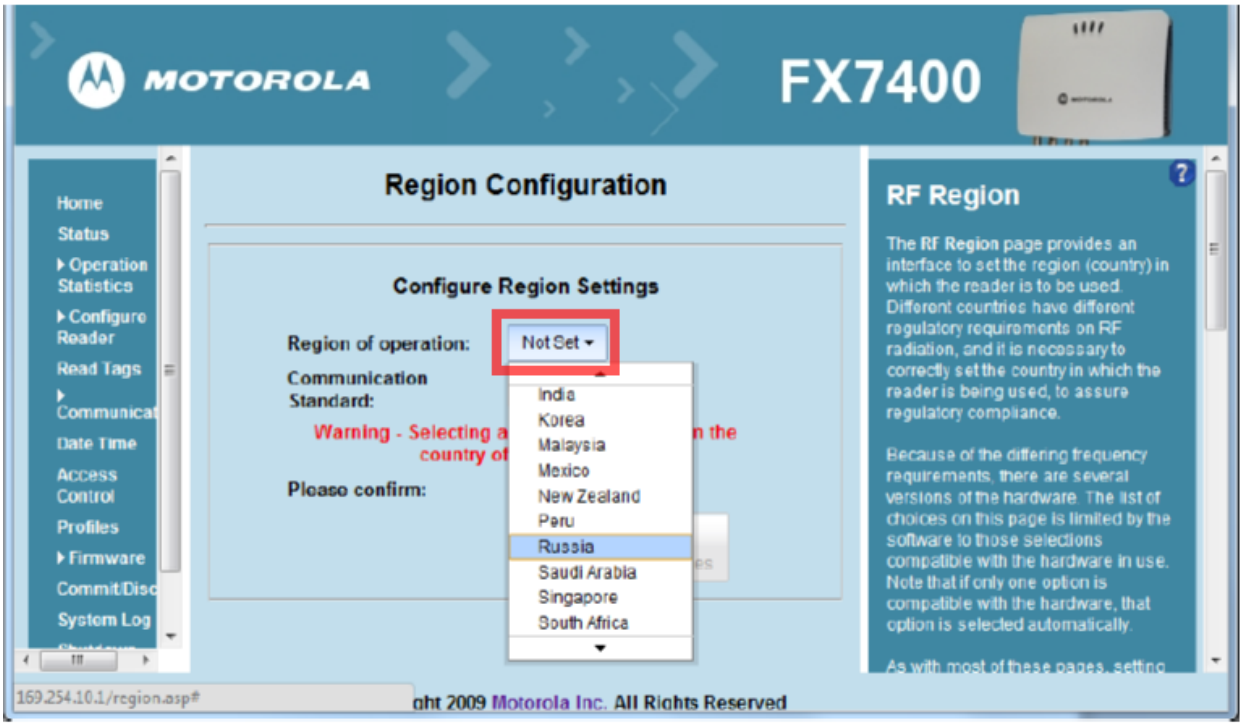


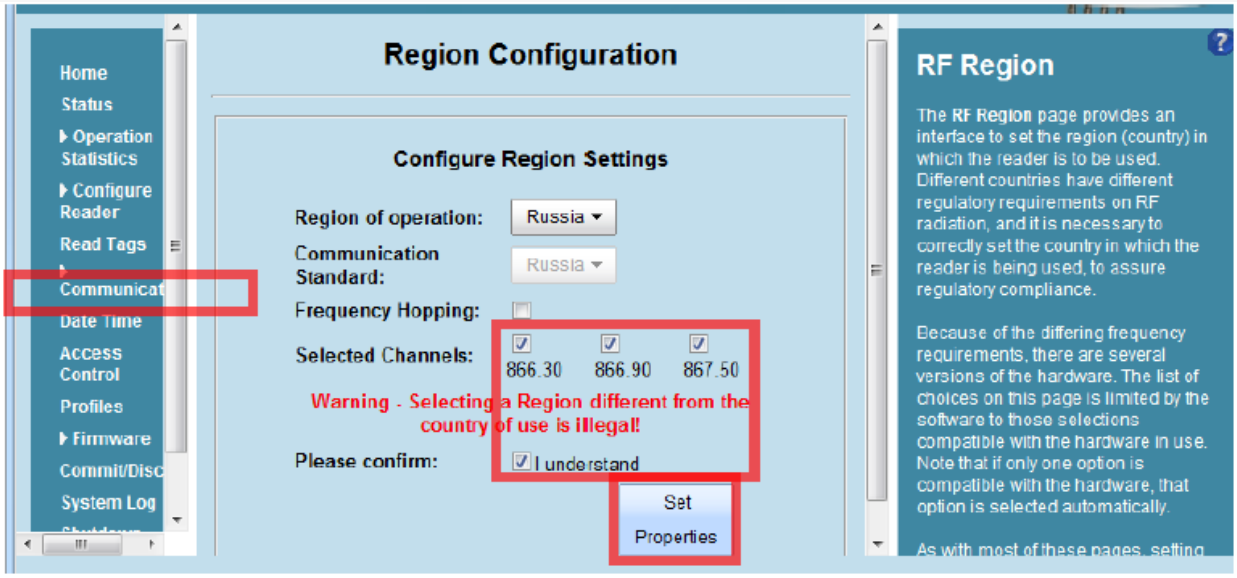
Перед началом работы со считывателем следует залогиниться в административную панель считывателя, используя интернет-браузер, и произвести настройку региона и диапазона используемых частот (логин

и пароль по умолчанию: «admin» и «change»):



Далее следует выставить регион «Russia» и соответствующие частоты:





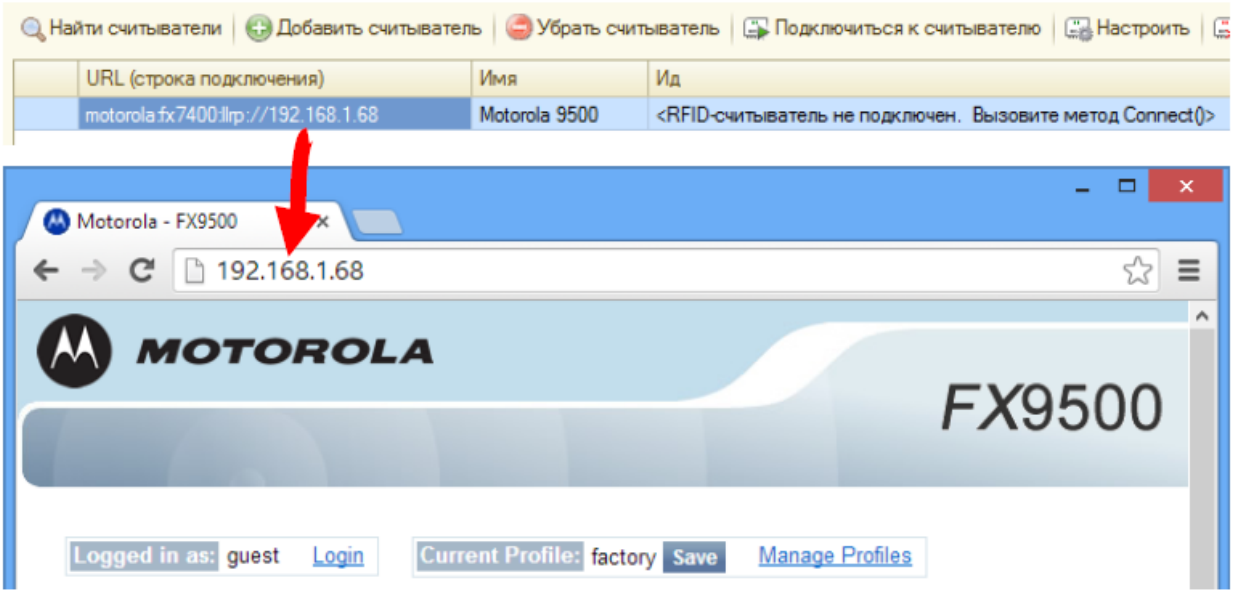
Следует проверить, что LLRP включен (если не включен — включить «Enable LLRP»):

После всех шагов идем в «Commit/Discard» и нажимаем «Commit» (применить изменения):

Установка и настройка Motorola FX9500

Установка считывателя Motorola FX9500 заключается в подключении его к питанию и сети Ethernet. Motorola FX9500 не работает через PoE и требует розетки и отдельного блока питания.

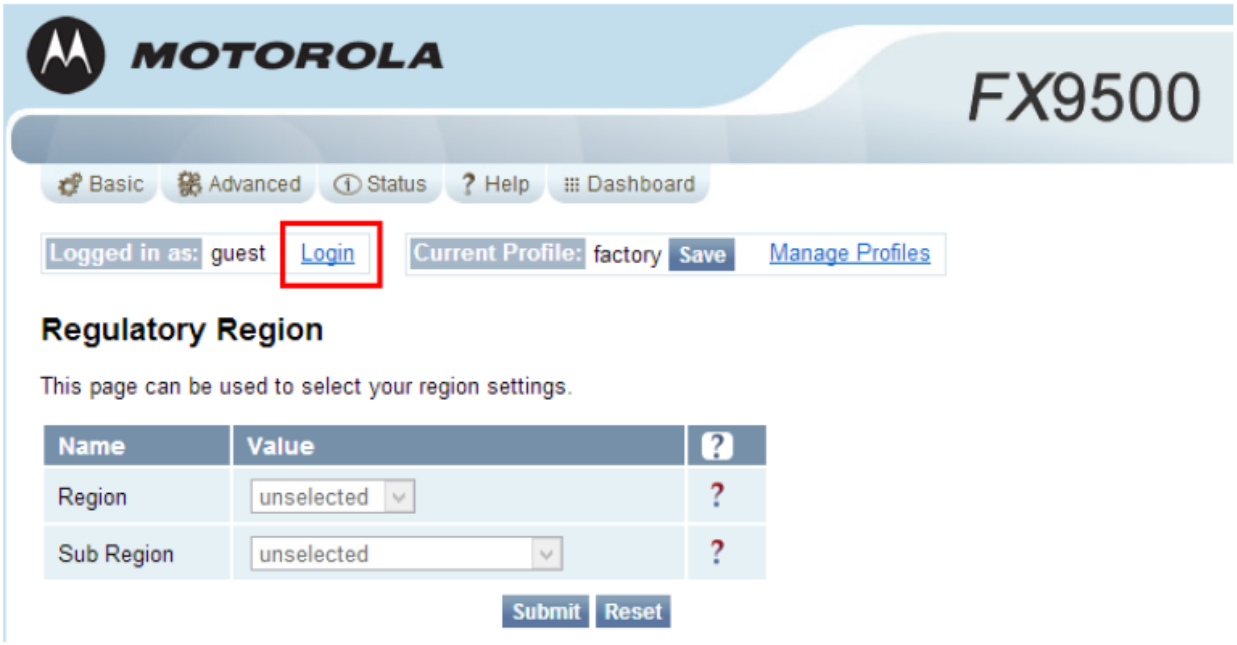
Считыватель Motorola FX9500 настраивается через веб-консоль из обычного браузера. Чтобы открыть консоль настройки считывателя, нужно вбить в адресной строке браузера его IP. Узнать IP можно поиском считывателя при помощи демообработки, которая поставляется вместе с компонентой:

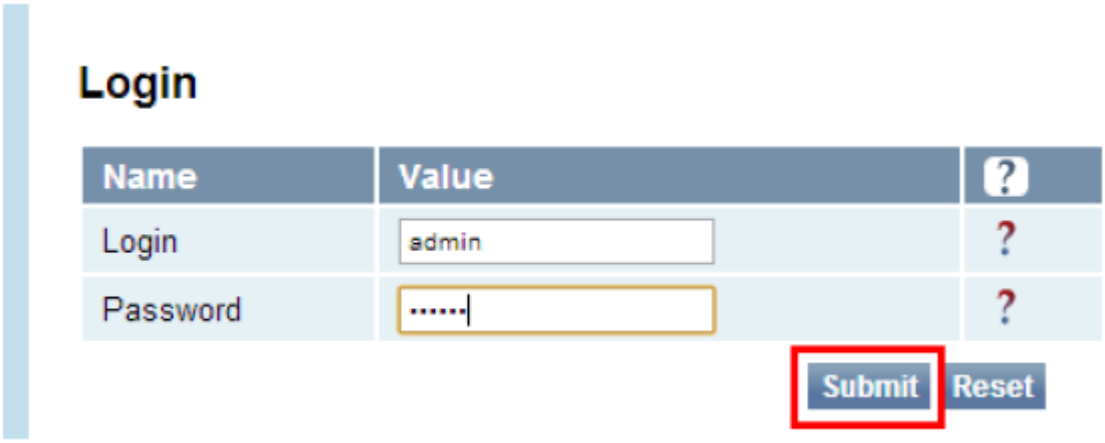


Предварительная настройка Motorola FX9500

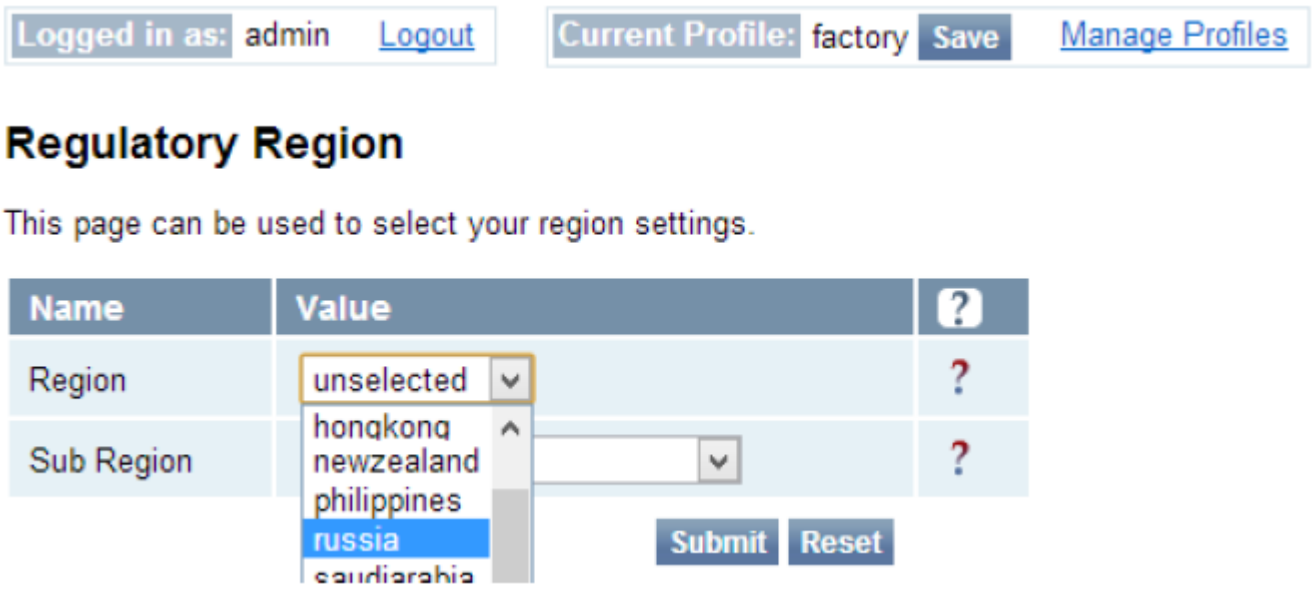
Перед тем, как можно будет прочесть хотя бы одну метку, в новом считывателе следует установить регион использования:

Чтобы поля стали доступны для изменения, следует залогиниться как администратор (логин и пароль по умолчанию: «admin» и «change»):

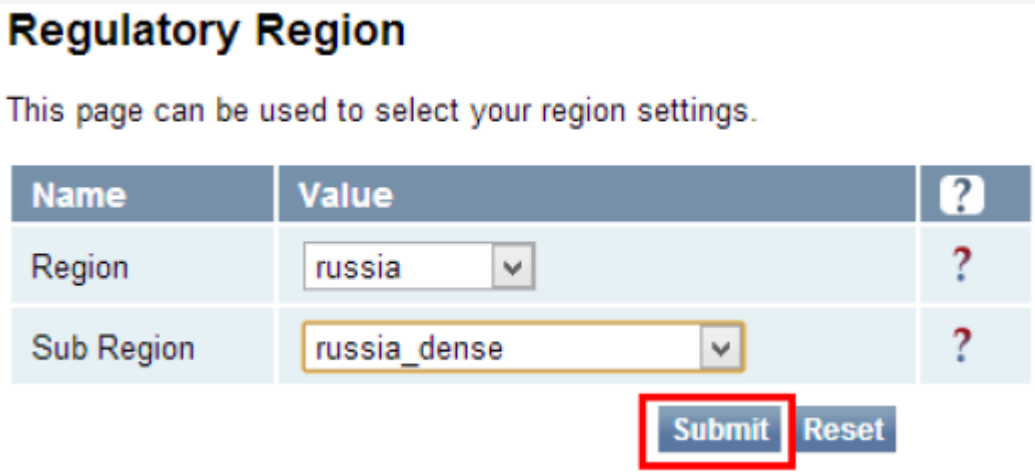




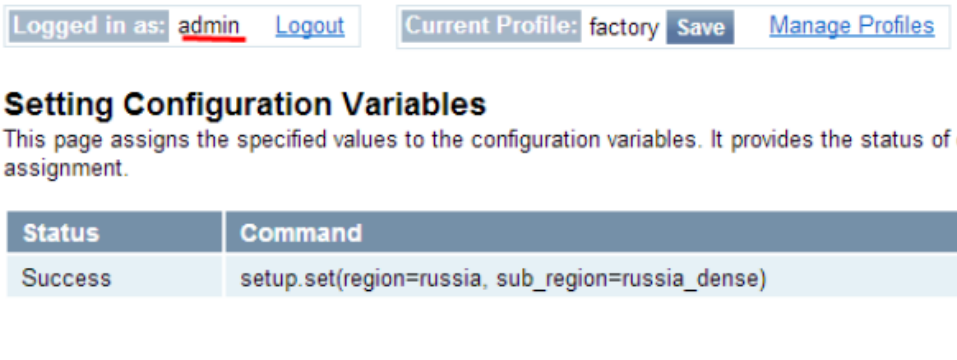
Залогинившись как админ, можно войти в «Set Regulatory Mode (Region)» и указать страну:



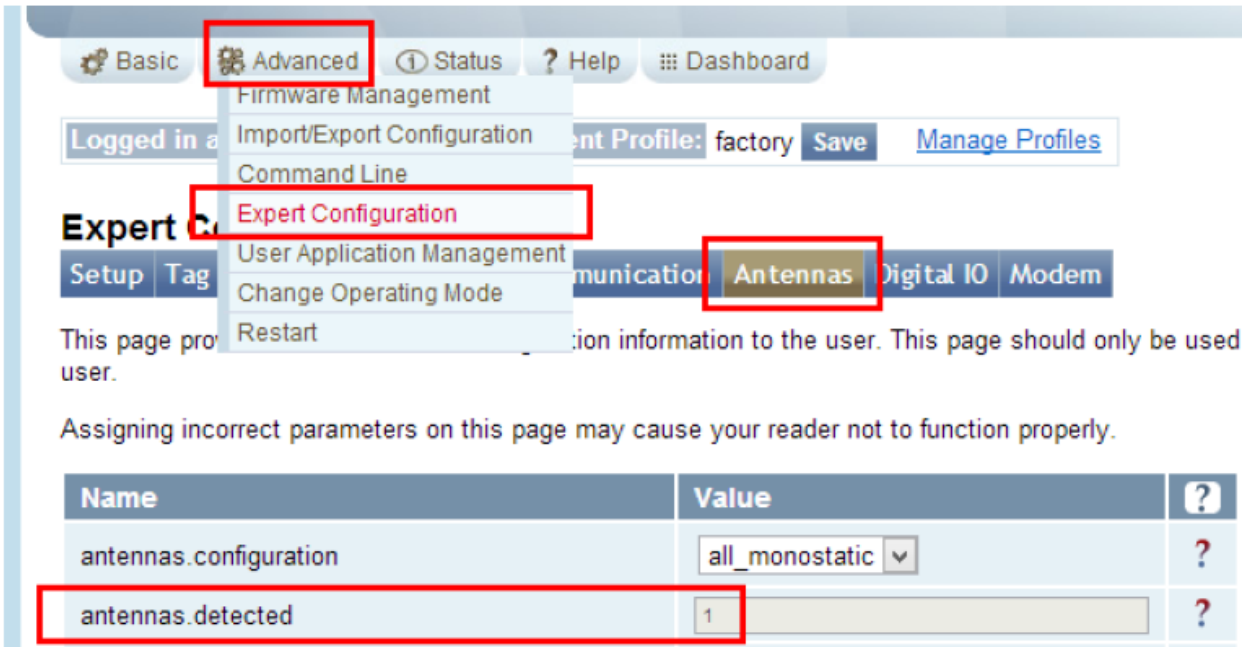
Выбрать «russia», «russia_dense» и нажать «Submit»:



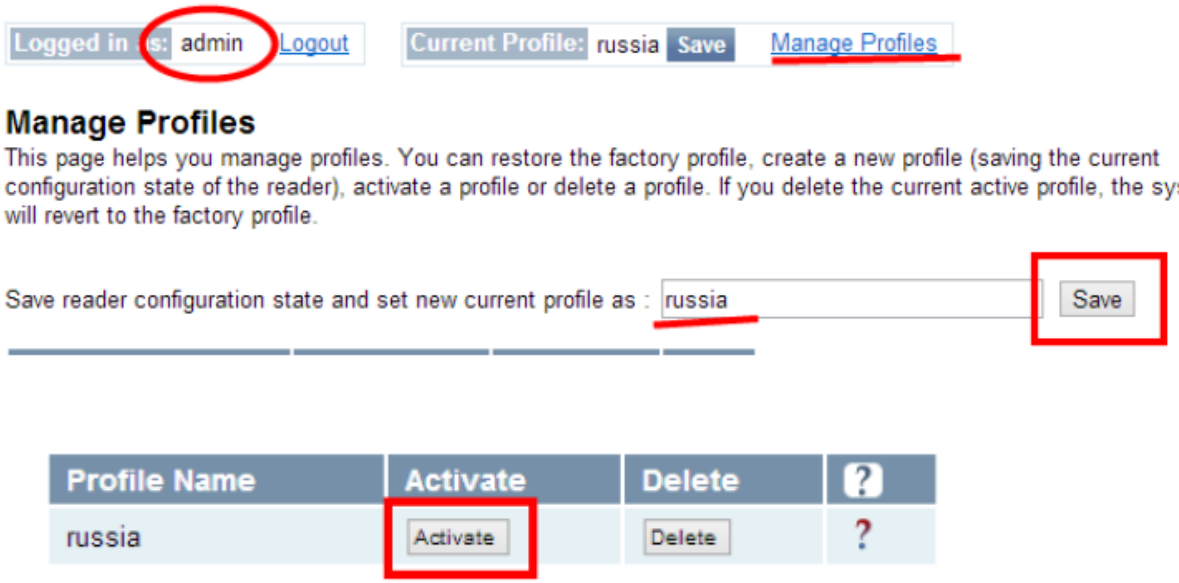
Результатом должно быть такое окно:



Только после данной настройки считыватель будет считывать метки. Если же этого не происходит, следует проверить кабель антенны. Для этого необходимо войти в меню «Advanced > Expert Configuration > Antennas»:



Далее нужно создать новый профиль настроек (например, «russia»), сохранить его и сделать профилем по умолчанию:



Только после этого настройку можно считать оконченной.

> Если в считывателе Motorola FX9500 не сохранить выбранные настройки в профиль (кнопки «Save», «Activate»), то после выключения/включения питания все выполненные настройки сбросятся!

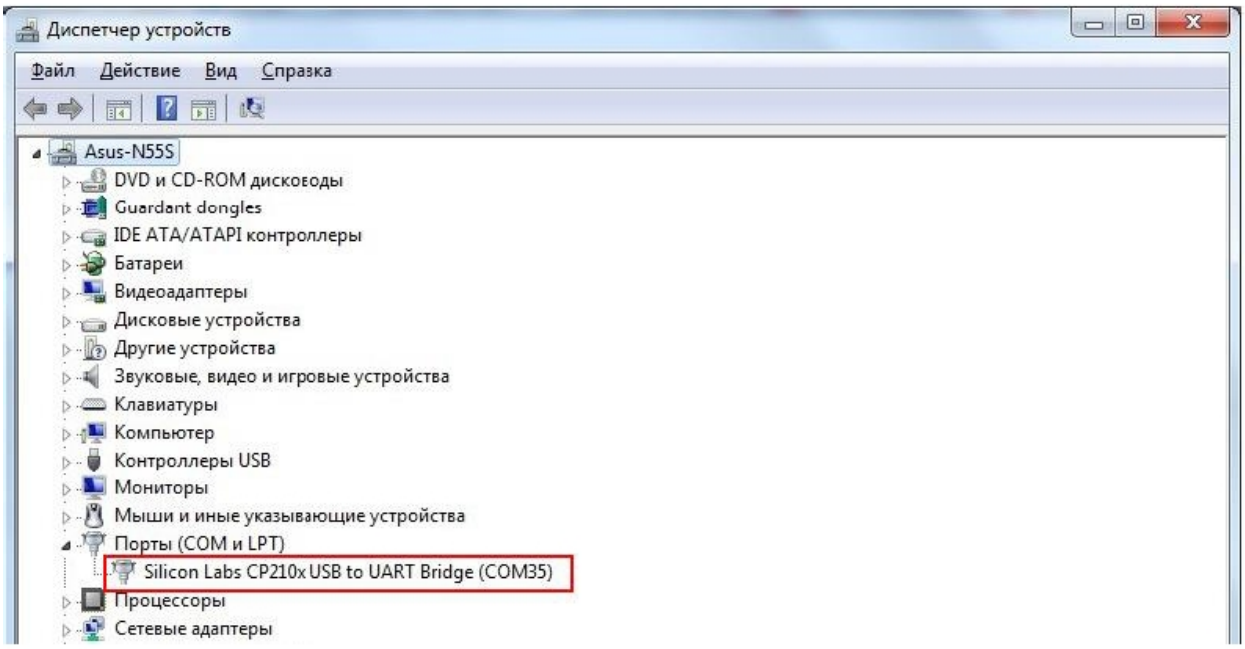
Установка и настройка RoyalRay RRU9809USBL

Установка считывателя RoyalRay RRU9809USBL заключается в подключении его к USB порту компьютера.

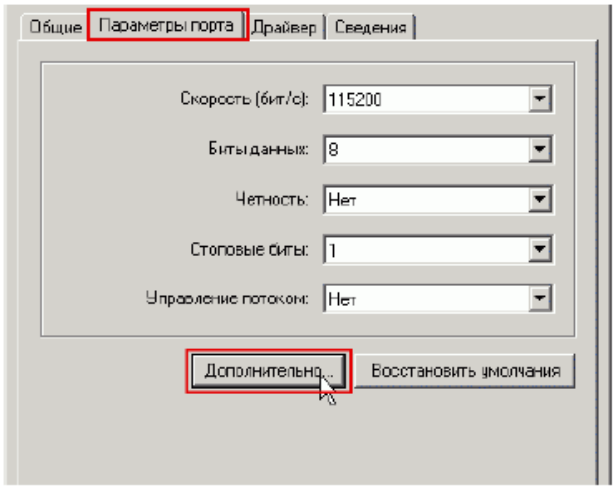
Далее следует загрузить и установить драйвер устройства в соответствии с разрядностью ОС.

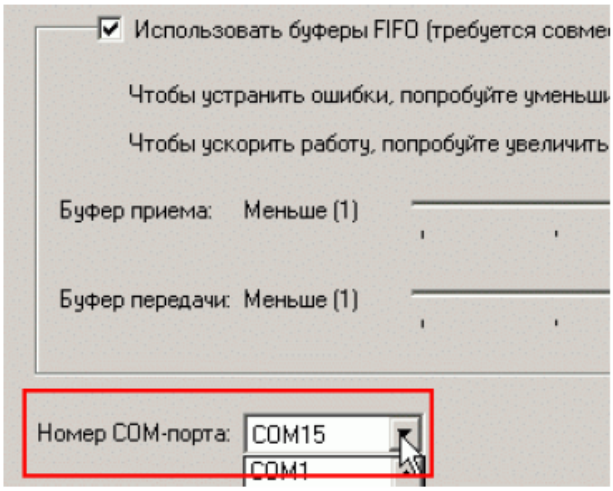
В случае возникновения проблем при подключении к устройству, необходимо проверить номер COM-порта подключённого устройства.

Для этого в «Диспетчер устройств» в группе «Порты (COM и LPT)» найти устройство Silicon Labs CP210x USB to UART Bridge и проверить, что устройству назначен COM порт в диапазоне от COM1 до COM9.



Если назначен больший номер, необходимо войти в «Свойства"-> «Параметры порта» -> «Дополнительно» -> «Номер COM-порта» и выбрать свободный, в диапазоне от COM1 до COM9.





URL для подключения к ридеру:

В зависимости от заданного COM-порта: от RoyalRay:COM1 до RoyalRay:COM9

Не нашли что искали?

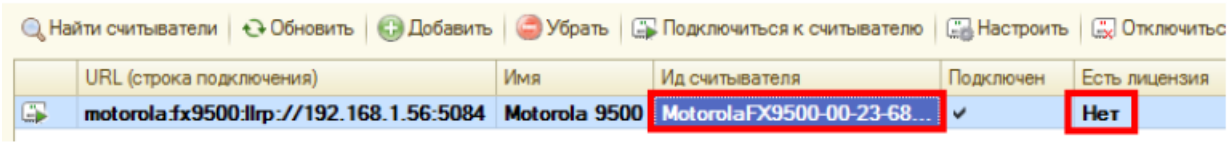
Задать вопрос в техническую поддержку

Лицензирование Wonderfid™ Link

Последние изменения: 2024-03-26

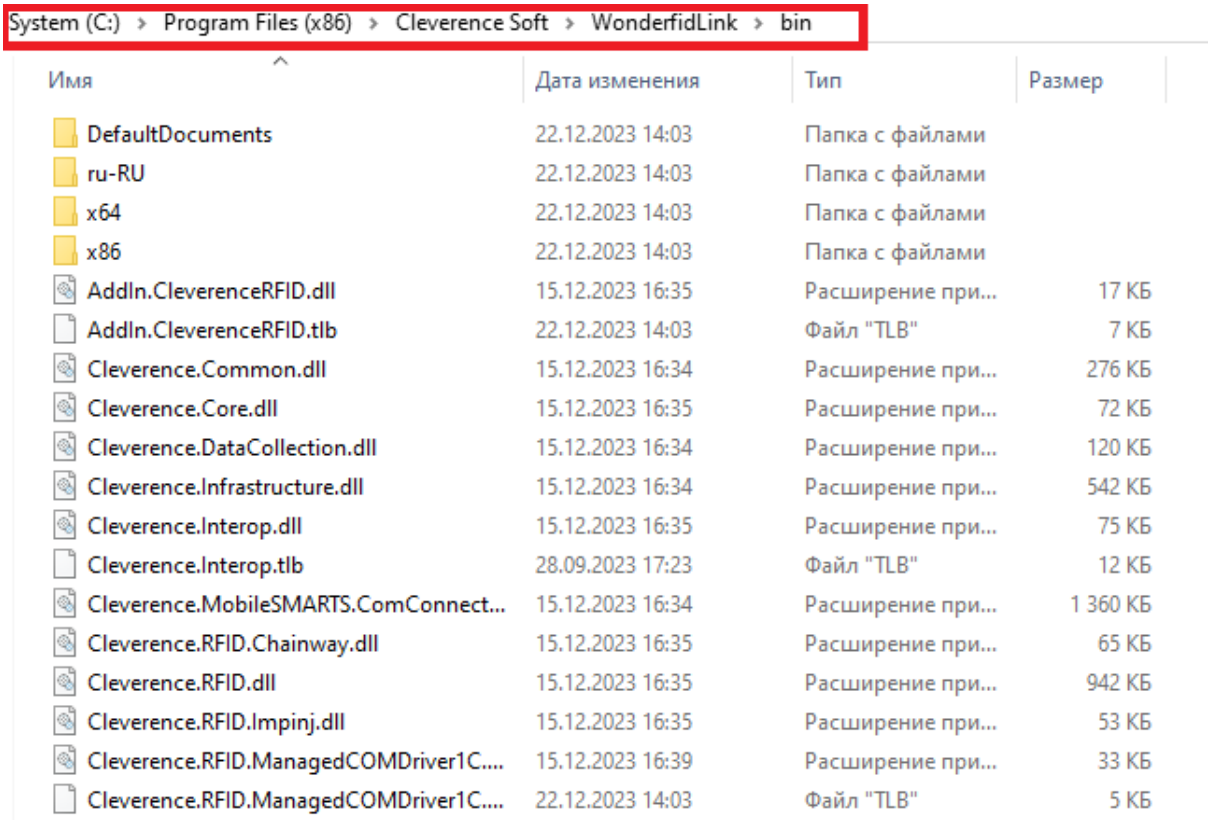
Лицензии «Wonderfid™ Link» приобретаются отдельно для каждого конкретного экземпляра RFID считывателя. Например, если у вас 5 считывателей модели Motorola FX7400, то вам следует приобрести 5 лицензий продукта Wonderfid™ Link.

Лицензии выдаются отдельно для каждого считывателя в соответствии с его идентификационным номером (ID):



Полученный код (например, «MotorolaFX9500-00-23-68-F2-82-F2») следует прислать в отдел продаж «Клеверенс» (sales@cleverence.ru). В ответ вы получите письмо с вложением: файл защиты лицензии с длинным названием, похожим на «license_CleverenceRFID_MotorolaFX9500-00-23-68-F0-89-F4_(30.07.2013_18-31) (МояФирма, счет).xml».

Этот файл следует разместить в папке установки продукта (например, «C:\Program Files (x86)\Cleverence Soft\WonderfidLink\bin») и лицензия тут же должна примениться:



Если по какой-то причине лицензия не работает, следует сначала посмотреть содержимое лога ошибок (см. CleverenceRFID_log.txt) или обратиться в техподдержку «Клеверенс».

Не нашли что искали?

Задать вопрос в техническую поддержку

Справочник разработчика «Wonderfid™ Link»

Последние изменения: 2024-03-26

Cleverence.RFID.Api

Предоставляет API для работы со стационарными RFID-считывателями.

Библиотеки (Libraries)
Содержит методы и перечисления, специфичные для библиотечного применения.

Версия (Version)
Возвращает версию компоненты.

Язык (Culture)
Возвращает или устанавливает текущую локализацию компоненты.

ВиртуальныйРежим (VirtualMode)
Возвращает настройки виртуального режима работы.

LookupTagParams (LookupTagParams)
Возвращает настройки проведения инвентаризации (слежения за метками) для всех RFID-считывателей. При этом собственные настройки конкретных считывателей могут добавлять/перекрывать общие настройки.

ФильтрыEPC (EpcFilterValues)
Возвращает коды фильтров EPC.

AFI (AFI)
Возвращает коды AFI/ASF (Application Family Identifier и Application Sub Family коды [ISO15961]), отражающее сферу применения (индустрию) объекта, на который нанесена метка.

УИИзБиблиотечногоКода (UIIfromISIL)
Создает экземпляр библиотечный вариант UII на основе переданных аргументов.
Синтаксис: УИИзБиблиотечногоКода (<isil>, <itemIdentifier>)

Имя параметра	Описание
isil	ISIL библиотеки-владельца или null.
itemIdentifier	Номер библиотечного объекта, уникальный в рамках конкретной библиотеки.

Возвращает: Экземпляр UII согласно стандарта ISO 28560.

УИИзБиблиотечногоКода (UIIfromISIL)
Создает экземпляр библиотечный вариант UII на основе переданных аргументов.
Синтаксис: УИИзБиблиотечногоКода (<isil>, <itemIdentifier>, <afi>)

Имя параметра	Описание
isil	ISIL библиотеки-владельца или null.
itemIdentifier	Номер библиотечного объекта, уникальный в рамках конкретной библиотеки.
afi	Код применения для объекта (важен для учета выдачи/возврата).

Возвращает: Экземпляр UII согласно стандарта ISO 28560.

СоздатьБиблиотечныйОбъект (CreateLibraryItem)
Создает пустой экземпляр набора записей о библиотечном объекте.
Синтаксис: СоздатьБиблиотечныйОбъект(), метод не принимает агрументов.
Возвращает: Пустой экземпляр набора записей о библиотечном объекте.

ОтключитьВсеСчитыватели (DisconnectAllReaders)
Освобождает все существующие подключения в рамках библиотеки.
Синтаксис: ОтключитьВсеСчитыватели(), метод не принимает агрументов.

НайтиСчитыватели (LookupReaders)
Производит поиск и возвращает список RFID-считывателей в локальной подсети.

> **Функция производит поиск считывателей только в локальных подсетях, т.е. в диапазонах IP-адресов: (192.168.0.1 – 192.168.248.255), (172.16.0.1 - 172.16.240.255) и (10.0.0.1 - 10.255.255.255).**

Синтаксис: НайтиСчитыватели(), метод не принимает агрументов.
Возвращает: Коллекция объектов типа «RfidReader (RfidReader)»

СчитывательЛицензирован (IsReaderLicensed)
Возвращает признак того, есть ли на данный считыватель лицензия.
Синтаксис: СчитывательЛицензирован (<readerId>)

ПолучитьСчитыватель (GetReader)
Получает существующий или создает новый RFID-считыватель по указанному URL.
Синтаксис: ПолучитьСчитыватель (<url>)

Имя параметра	Описание
url	URL считывателя с указанием типа подключения, адреса и порта.

ПодключитьСчитыватель (ConnectToReader)
Выполняет подключение к RFID-считывателю по указанному URL (с возможностью чтения/записи, но без возможности управления параметрами считывателя, см. «ПодключитьСчитыватель (ConnectToReader)»).
Синтаксис: ПодключитьСчитыватель (<url>)

Имя параметра	Описание
url	URL для подключения с указанием типа подключения, адреса и порта.

Возвращает: Объект, отвечающий за работу с RFID-считывателем, к которому было выполнено подключение.

ПодключитьСчитывательПодПаролем (ConnectToReaderWithPassword)
Выполняет подключение к RFID-считывателю по указанному URL (с возможностью как чтения/записи, так и управления параметрами считывателя).
Синтаксис: ПодключитьСчитывательПодПаролем (<url>, <userName>, <password>)

Имя параметра	Описание
url	URL для подключения с указанием типа подключения, адреса и порта.
userName	Имя пользователя для подключения.
password	Пароль пользователя.

Возвращает: Объект, отвечающий за работу с RFID-считывателем, к которому было выполнено подключение.

ВыбратьМетку (FetchTag)

Вынимает из очереди считанных меток данные метки (меток) с указанным Tag ID сразу со всех RFID-считывателей, на которых в рамках библиотеки было запущено чтение меток.

Т.к. метки вынимаются из очереди, второй и последующие вызовы метода с одним и тем же Tag ID могут вернуть пустой результат.

Примеры запросов

```
// выбрать метку с указанным Tag Id:
FetchTag("303000181CE257587E9C000");
// выбрать метку с указанным Tag Id, прочитанную командой с указанным кодом:
FetchTag("EEFE0574-B63A-4AEB-B4D2-1986B9D74930@303000181CE257587E9C000");
// выбрать метку с указанным Tag Id, прочитанную считывателем с указанным url:
FetchTag("motorola:xr480:llrp://10.10.0.17@303000181CE257587E9C000");
```

Синтаксис: ВыбратьМетку (<fetchQuery>)

Имя параметра	Описание
fetchQuery	Строка с запросом интересующих меток. В запросе через знак @ могут указываться id команды, url считывателя и Tag ID метки.

Возвращает: Данные метки (меток), которые были вынуты из очереди считанных меток.

ВыбратьМетки (FetchTags)

Вынимает из очереди считанных меток данные всех меток сразу со всех RFID-считывателей, на которых в рамках библиотеки было запущено чтение меток.
Т.к. метки вынимаются из очереди, второй и последующие вызовы метода могут вернуть пустой результат.
Синтаксис: ВыбратьМетки(), метод не принимает агрументов.
Возвращает: Данные меток, которые были вынуты из очереди считанных меток.

ВыбратьМетки (FetchTags)

Вынимает из очереди считанных меток данные всех меток сразу со всех RFID-считывателей, на которых в рамках библиотеки было запущено чтение меток.

Т.к. метки вынимаются из очереди, второй и последующие вызовы метода могут вернуть пустой результат.

Примеры запросов

```
// выбрать метки с указанным Tag Id, прочитанные всеми считывателями:
FetchTag("303000181CE257587E9C000");
// выбрать метки, прочитанные командой с указанным кодом на всех считывателях:
FetchTag("EEFE0574-B63A-4AEB-B4D2-1986B9D74930");
// выбрать метки, прочитанные командой с указанным кодом на считывателе с указанным url:
FetchTag("EEFE0574-B63A-4AEB-B4D2-1986B9D74930@motorola:xr480:llrp://10.10.0.17");
```

Синтаксис: ВыбратьМетки (<fetchQuery>)

Имя параметра	Описание
fetchQuery	Строка с запросом интересующих меток. В запросе через знак @ могут указываться id команды, url считывателя и Tag ID метки.

Возвращает: Данные меток, которые были вынуты из очереди считанных меток.

EPCизSGTIN (EPCfromSGTIN)
Создает экземпляр SGTIN-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизSGTIN (<company>, <item>, <filterValue>, <serial>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
item	Код товара согласно каталога компании.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
serial	Серийный номер экземпляра товара.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизSGTIN (EPCfromSGTIN)
Создает экземпляр SGTIN-варианта EPC на основе кода компании и кода товара. Серийный номер будет сгенерирован компонентой при записи в метку.
Синтаксис: EPCизSGTIN (<company>, <item>, <filterValue>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
item	Код товара согласно каталога компании.
filterValue	Filter Value кода для указания типа упаковки, для которо предназначен данный EPC.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизSGTIN (EPCfromSGTIN)
Создает экземпляр SGTIN-варианта EPC на основе кода компании и кода товара. Серийный номер будет сгенерирован компонентой при записи в метку.
Синтаксис: EPCизSGTIN (<company>, <item>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
item	Код товара согласно каталога компании.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизШК (EPCfromBarcode)

Создает экземпляр SGTIN-варианта EPC на основе штрихкода.
Синтаксис: EPCизШК (<barcode>, <filterValue>, <serial>)

Имя параметра	Описание
barcode	Строка со штрихкодом EAN8, EAN13, ISBN, ISSN, UPC или EAN128.
filterValue	Filter Value кода для указания типа упаковки, для которо предназначен данный EPC.
serial	Серийный номер экземпляра товара.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизШК (EPCfromBarcode)
Создает экземпляр SGTIN-варианта EPC на основе штрихкода.
Синтаксис: EPCизШК (<barcode>, <filterValue>)

Имя параметра	Описание
barcode	Строка со штрихкодом EAN8, EAN13, ISBN, ISSN, UPC или EAN128.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизШК (EPCfromBarcode)
Создает экземпляр SGTIN-варианта EPC на основе штрихкода.
Синтаксис: EPCизШК (<barcode>)

Имя параметра	Описание
barcode	Строка со штрихкодом EAN8, EAN13, ISBN, ISSN, UPC или EAN128.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизGDTI (EPCfromGDTI)
Создает экземпляр GDTI-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизGDTI (<company>, <documentType>, <documentSerial>, <filterValue>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
documentType	Числовой код типа документа.
documentSerial	Серийный номер экземпляра документа.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизGDTI (EPCfromGDTI)
Создает экземпляр GDTI-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизGDTI (<company>, <documentType>, <documentSerial>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
documentType	Числовой код типа документа.
documentSerial	Серийный номер экземпляра документа.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизSSCC (EPCfromSSCC)
Создает экземпляр SSCC-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизSSCC (<company>, <extension>, <filterValue>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
extension	Числовой номер паллеты без префикса кода компании.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизSSCC (EPCfromSSCC)
Создает экземпляр SSCC-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизSSCC (<company>, <extension>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
extension	Числовой номер паллеты без префикса кода компании.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCизGRAI (EPCfromGRAI)
Создает экземпляр GRAI-варианта EPC на основе переданных аргументов.
Синтаксис: EPCизGRAI (<company>, <assetType>, <serial>)

Имя параметра	Описание
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
company	Код компании, зарегистрированной в GS1.
assetType	Числовой номер типа оборачиваемой тары (назначается компанией самостоятельно).
serial	Сирийный номер экземпляра оборачиваемой тары.

Возвращает: Экземпляр EPC согласно стандарта GS1.

EPCfromGIAI (EPCfromGIAI)
Создает экземпляр GIAI-варианта EPC на основе переданных аргументов.
Синтаксис: EPCfromGIAI (<company>, <assetReference>)

Имя параметра	Описание
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
company	Код компании, зарегистрированной в GS1.
assetReference	Числовой номер индивидуального имущества (основного средства).

Возвращает: Экземпляр EPC согласно стандарта GS1.

UIIизDI (UIIfromDI)
Создает экземпляр UII по переданному DI его строковому значению.
Синтаксис: UIIизDI (<di>, <value>)

Имя параметра	Описание
di	
value	

НоваяМетка (CreateTag)
Создает экземпляр метки на основе указанного Tag ID.
Синтаксис: НоваяМетка (<tagId>)

Имя параметра	Описание
tagId	Tag ID метки.

Возвращает: Созданная метка.

НоваяМетка (CreateTag)
Создает экземпляр метки на основе указанного EPC или UII.
Синтаксис: НоваяМетка (<oi>)

Имя параметра	Описание
epc	EPC или UII метки.

Возвращает: Созданная метка.
СоздатьБиблиотечныйОбъект (CreateLibraryItem)
Создает и заполняет набор записей о библиотечном объекте на основе данных из USER-банка памяти RFID-метки.
Синтаксис: СоздатьБиблиотечныйОбъект (<bank>)

Имя параметра	Описание
bank	Экземпляр USER-банка памяти RFID-метки.

Возвращает: Набор записей о библиотечном объекте, заполненный на основе переданного банка.

Cleverence.RFID.RfidReader

Содержит методы по работе со стационарным RFID-считывателем.

LookupTagParams (LookupTagParams)
Возвращает настройки проведения инвентаризации (слежения за метками) для данного RFID-считывателя. Они могут добавлять/перекрывать общие настройки для всех считывателей, указанные в «LookupTagParams (LookupTagParams)».

Имя (DisplayName)
Возвращает отображаемое имя RFID-считывателя.

Ид (Id)
Возвращает идентификатор RFID-считывателя.

Url (Url)
Возвращает URL до RFID-считывателя.

Подключен (IsConnected)
Возвращает true (Истина), если подключение к RFID-считывателю активно.

Авторизован (IsLoggedIn)
Возвращает true (Истина), если подключение к RFID-считывателю произошло под логином/паролем и/или разрешает изменять настройки.

ИдетИнвентаризация (LookupTagsInProgress)
Возвращает true (Истина), если считыватель в настоящее время выполняет инвентаризацию.

РазрешатьПовторныеЧтения (AllowRepetitiveReads)
Возвращает или устанавливает флаг, указывающий библиотеке, следует ли ему при чтении возвращать повторные считывания. Если флаг не выставлен - все события и данные об индивидуальных считываний приходят раздельно. Если флаг выставлен - все события и данные группируются/объединяются по Tag ID и содержимым банков.

Подключить (Connect)
Выполняет подключение к RFID-считывателю (с возможностью чтения/записи, но без возможности управления параметрами считывателя, см. «ПодключитьПодПаролем (ConnectWithPassword)»).
Синтаксис: Подключить(), метод не принимает агрументов.

ПодключитьПодПаролем (ConnectWithPassword)
Выполняет подключение к RFID-считывателю с возможностью как чтения/записи, так и управления параметрами считывателя).
Синтаксис: ПодключитьПодПаролем (<userName>, <password>)

Имя параметра	Описание
userName	Имя пользователя для подключения.
password	Пароль пользователя.

ВыбратьМетку (FetchTag)
Вынимает из очереди считанных меток данные метки (меток) с указанным Tag ID.

> Т.к. метки вынимаются из очереди, второй и последующие вызовы метода с одним и тем же Tag ID могут вернуть пустой результат. Поведение зависит от флага, разрешающего повторные чтения. При повторных чтениях метки могут снова оказаться в очереди.

Синтаксис: ВыбратьМетку (<tagid>)

Имя параметра	Описание
tagid	Tag ID интересующих меток.

Возвращает: Данные метки (меток), которые были вынуты из очереди считанных меток.

ВыбратьМетку (FetchTag)
Вынимает из очереди считанных меток данные метки (меток) с указанным Tag ID.

> Т.к. метки вынимаются из очереди, второй и последующие вызовы метода с одним и тем же Tag ID могут вернуть пустой результат. Поведение зависит от флага, разрешающего повторные чтения. При повторных чтениях метки могут снова оказаться в очереди.

Синтаксис: ВыбратьМетку (<tagid>, <commandId>)

Имя параметра	Описание
tagid	Tag ID интересующих меток.
commandId	Id команды, при помощи которой была записана или прочитана интересующая метка.

Возвращает: Данные метки (меток), которые были вынуты из очереди считанных меток.

ВыбратьМетки (FetchTags)
Вынимает из очереди считанных меток данные всех меток.

> Т.к. метки вынимаются из очереди, второй и последующие вызовы метода могут вернуть пустой результат. Поведение зависит от флага, разрешающего повторные чтения. При повторных чтениях метки могут снова оказаться в очереди.

Синтаксис: ВыбратьМетки(), метод не принимает аргументов.
Возвращает: Данные меток, которые были вынуты из очереди считанных меток.

ВыбратьМетки (FetchTags)
Вынимает из очереди считанных меток данные всех меток.

ПолучитьВозможности (GetCapabilities)
Получает и возвращает конфигурацию RFID-считывателя.
Синтаксис: ПолучитьВозможности(), метод не принимает агрументов.
Возвращает: Объект типа RfidReaderCapabilities.

ИнвентаризоватьМетки|ПрочестьМетки (LookupTags)
Выполняет поиск и чтение Tag ID меток в радиусе видимости считывателя в течение определенного времени.

> Очищает очередь считанных меток.

Синтаксис: ИнвентаризоватьМетки|ПрочестьМетки (<readTime>)

Имя параметра	Описание
commandId	Id команды, при помощи которой были записаны или прочитаны интересующие метки.

Возвращает: Коллекция объектов типа RfidTag.

ИнвентаризоватьМетки|ПрочестьМетки (LookupTags)
Выполняет поиск и чтение Tag ID меток в радиусе видимости считывателя в течение определенного времени.

> Очищает очередь считанных меток.

Синтаксис: ИнвентаризоватьМетки|ПрочестьМетки (<readTime>, <password>, <readTid>, <readUser>, <readReserved>)

Имя параметра	Описание
readTime	Количество времени в миллисекундах, в течение которого считывателю следует искать метки.

Возвращает: Коллекция объектов типа RfidTag.

НачатьИнвентаризацию|НачатьЧтение (LookupTagsBegin)
Начинает поиск и чтение Tag ID меток в радиусе видимости считывателя в течение определенного времени. Функция выполняется асинхронно и возвращает управление вызывающей стороне сразу после начала чтения. Считанные метки приходят в событии «TagRead (TagRead)».

> Очищает очередь считанных меток.

Синтаксис: НачатьИнвентаризацию|НачатьЧтение (<readTime>)

Имя параметра	Описание
readTime	Количество времени в миллисекундах, в течение которого считывателю следует искать метки.
readReserved	Читать ли RESERVED банк меток.
readTid	Читать ли TID банк меток.
readUser	Читать ли USER банк меток.
password	Пароль доступа, может оказаться необходим для чтения дополнительных банков (RESERVED, TID или USER)

Возвращает: Строку с Id команды, в рамках которой будет выполняться чтение.

НачатьИнвентаризацию|НачатьЧтение (LookupTagsBegin)
Начинает поиск и чтение Tag ID меток в радиусе видимости считывателя в течение определенного времени. Функция выполняется асинхронно и возвращает управление вызывающей стороне сразу после начала чтения. Считанные метки приходят в событии «TagRead (TagRead)».

> **Очищает очередь считанных меток.**

Синтаксис: НачатьИнвентаризацию|НачатьЧтение (<readTime>, <password>, <readTid>, <readUser>, <readReserved>)

Возвращает: Строку с Id команды, в рамках которой будет выполняться чтение.

ОкончитьИнвентаризацию|ОкончитьЧтение (LookupTagsEnd)
Прерывает работу по чтению меток, инициированную вызовом функции «НачатьИнвентаризацию|НачатьЧтение (LookupTagsBegin)».
Возвращает всё, что было считано (не только из очереди считанных меток, а вообще все метки), и очищает очередь считанных меток.
Синтаксис: ОкончитьИнвентаризацию|ОкончитьЧтение(), метод не принимает аргументов.

ПрочестьБанкEPCUII (ReadEPCUII)
Производит чтение EPC/UII-банка (банка 01) первой попавшейся RFID-метки с указанным значением Tag ID, с указанием пароля на доступ (Access Password, хранящийся в банке 00 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ПрочестьБанкEPCUII (<tagId>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID метки для чтения.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Объект, отражающий данные банка EPC/UII метки, либо содержащий описание ошибки.

ЗаписатьEPCUII (WriteEPCUII)
Производит запись в EPC/UII-банк (банк 01) во все RFID-метки с указанным значением Tag ID, с указанием пароля на доступ (Access Password, хранящийся в банке 00 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ЗаписатьEPCUII (<tagId>, <epcuii>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID метки для чтения.
epcuii	Записываемый EPC/UII - электронный код товара или уникальный идентификатор объекта.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

ЗаписатьEPCUIIпоTID (WriteEPCUIIforTID)
Производит запись в EPC-банк (банк 01) первой попавшейся RFID-метки с указанным значением Tag ID и содержимым банка TID (уникальный номер чипа, который, в отличие от Tag ID действительно всегда уникален). С указанием пароля на доступ (Access Password, хранящийся в банке 00 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ЗаписатьEPCUIIпоTID (<tagId>, <tid>, <epcuii>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID нужной метки для записи.
tid	Содержимое банка TID нужной метки для записи.
epcuii	Записываемый EPC/UII - электронный код товара или уникальный идентификатор объекта.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

ЗаписатьEPCUIIпоTID (WriteEPCUIIforTID)
Производит запись в EPC-банк (банк 01) первой попавшейся RFID-метки с указанным значением Tag ID и содержимым банка TID (уникальный номер чипа, который, в отличие от Tag ID действительно всегда уникален). С указанием пароля на доступ (Access Password, хранящийся в банке 00 RFID-метки). Если пароля нет, то следует указать 0. С указанием, следует ли блокировать возможность дальнейшей перезаписи значения EPC-банка. Если нужно заблокировать(залочить), то следует указать Истина.
Синтаксис: ЗаписатьEPCUIIпоTID (<tagId>, <tid>, <epcuii>, <accessPassword>, <_lock>)

Имя параметра	Описание
tagId	Tag ID нужной метки для записи.
tid	Содержимое банка TID нужной метки для записи.
epcuii	Записываемый EPC/UII - электронный код товара или уникальный идентификатор объекта.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

ПрочестьБанкRESERVED (ReadRESERVED)
Производит чтение RESERVED-банка (банка 00) первой попавшейся RFID-метки с указанным значением Tag ID, с указанием пароля на доступ (Access Password, хранящийся в банке 00 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ПрочестьБанкRESERVED (<tagId>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID метки для чтения.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Объект, отражающий данные банка TID метки, либо содержащий описание ошибки.

ПрочестьБанкTID (ReadTID)

Производит чтение TID-банка (банка 10) первой попавшейся RFID-метки с указанным значением Tag ID, с указанием пароля на доступ (Access Password, хранящийся в банке 10 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ПрочестьБанкTID (<tagId>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID метки для чтения.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Объект, отражающий данные банка TID метки, либо содержащий описание ошибки.

ReadTIDs (ReadTIDs)

Производит чтение TID-банка (банка 10) всех RFID-меток в поле видимости считывателя, с указанием пароля на доступ (Access Password, хранящийся в банке 10 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ReadTIDs (<readTime>, <accessPassword>)

Имя параметра	Описание
readTime	Количество времени в миллисекундах, в течение которого считывателю следует искать метки.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Коллекция меток, с прочитанным и заполненным банком TID.

ПрочестьБанкUSER (ReadUSER)

Производит чтение USER-банка (банка 11) первой попавшейся RFID-метки с указанным значением Tag ID, с указанием пароля на доступ (Access Password, хранящийся в банке 11 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ПрочестьБанкUSER (<tagId>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID метки для чтения.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Объект, отражающий данные банка USER метки, либо содержащий описание ошибки.

ReadUSERs (ReadUSERs)
Производит чтение USER-банка (банка 11) всех RFID-меток в поле видимости считывателя, с указанием пароля на доступ (Access Password, хранящийся в банке 11 RFID-метки). Если пароля нет, то следует указать 0.
Синтаксис: ReadUSERs (<readTime>, <accessPassword>)

Имя параметра	Описание
readTime	Количество времени в миллисекундах, в течение которого считывателю следует искать метки.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Возвращает: Коллекция меток, с прочитанным и заполненным банком USER (см. «USER (USER)»).

ВключитьАнтенну (EnableAntenna)
Активирует (включает) использование антенны с указанным номером (кодом).
Синтаксис: ВключитьАнтенну (<antennaId>)

Имя параметра	Описание
antennaId	Номер (код) антенны согласно Cleverence.RFID.RfidAntennaInfo.

ВыключитьАнтенну (DisableAntenna)
Деактивирует (выключает) использование антенны с указанным номером (кодом).
Синтаксис: ВыключитьАнтенну (<antennaId>)

Имя параметра	Описание
antennaId	Номер (код) антенны согласно Cleverence.RFID.RfidAntennaInfo.

УстановитьВходнуюМощностьДляАнтенны (SetTransmitPower)
Устанавливает входную мощность для антенны с указанным номером (кодом).
Синтаксис: УстановитьВходнуюМощностьДляАнтенны (<antennaId>, <powerLevel>)

Имя параметра	Описание
antennaId	Номер (код) антенны согласно Cleverence.RFID.RfidAntennaInfo.
powerLevel	Требуемая мощность в процентах от максимальной (от 1 до 100).

Возвращает: Новое значение входной мощности, подаваемой на антенну, в dBi.
Отключить (Disconnect)
Выполняет отключение от считывателя. Ничего не принимает и ничего не возвращает.
Синтаксис: Отключить(), метод не принимает аргументов.

Cleverence.RFID.RfidReaderCapabilities

Содержит информацию об оснащении и возможностях RFID-считывателю.

Антенны (Antennas)
Возвращает коллекцию описаний антенн RFID-считывателя.

Cleverence.RFID.RfidReaderCollection

Коллекция объектов типа «RfidReader (RfidReader)».

Количество (Count)
Возвращает количество элементов в списке.

Элемент (get_Item)
Возвращает элемент по указанному индексу.
Синтаксис: Элемент(<Индекс>)

Имя параметра	Описание
Индекс	Индекс элемента в вписке, от 0 до (Количество - 1).

Добавить (Add)
Добавляет в список новый элемент.
Синтаксис: Добавить(<Элемент>)

Удалить (Remove)
Удаляет из списка указанный элемент.
Синтаксис: Удалить(<Элемент>)

УдалитьПоИндексу (RemoveAt)
Удаляет из списка элемент по указанному индексу.
Синтаксис: УдалитьПоИндексу(<Индекс>)

Имя параметра	Описание
Индекс	Индекс элемента в вписке, от 0 до (Количество - 1).

Добавить (Add)
Добавляет в список новый элемент.
Синтаксис: Добавить(<Элемент>)

Удалить (Remove)
Удаляет из списка указанный элемент.
Синтаксис: Удалить(<Элемент>)

УдалитьПоИндексу (RemoveAt)
Удаляет из списка элемент по указанному индексу.
Синтаксис: УдалитьПоИндексу(<Индекс>)

Имя параметра	Описание
url	Url для подключения к RFID-считывателю.

Возвращает: Добавленный RFID-считыватель.

Cleverence.RFID.RfidTag

Содержит данные о RFID-метке на основании операции инвентаризации окружающих меток RFID-считывателем.

TagId (TagId)
Возвращает Tag ID метки 16-ричном представлении (строка в 24 символа).

Объект (Identity)
Возвращает значение EPC/UII метки (если метка закодирована в соответствии со стандартом EPCglobal или ISO), полученный на основании операции инвентаризации окружающих меток RFID-считывателем.

Считыватель (Reader)
Возвращает считыватель, при помощи которого была считана или записана данная метка.

Command (Command)
Возвращает команду, при помощи которой была считана или записана данная метка.

UrlСчитывателя (ReaderUrl)
Возвращает Url считывателя, при помощи которого была считана данная метка.

ExtendedFields (ExtendedFields)
Возвращает коллекцию расширенных свойств RFID-метки.

НомерАнтенны (AntennaId)
Возвращает номер (код) антенны, которая прочла метку с таким Tag ID.

Время (FirstTimeSeen)
Возвращает дату/время, в которое метка с таким Tag ID была увидена впервые (по часам компьютера, на котором работает Api)

Счетчик (SeenCount)
Возвращает, сколько раз такая метка была прочитана считывателем. Если при инвентаризации читались только Tag ID (и не читались дополнительные банки типа TID), то это сумма чтений всех меток с таким Tag ID.

RSSI (PeakRSSI)
Возвращает пиковое значение принятого уровня сигнала от метки в произвольных единицах от 0 до 255 (число).

Cleverence.RFID.RfidTagCollection

Коллекция объектов типа «RfidTag (RfidTag)».

Количество (Count)
Возвращает количество элементов в списке.

Элемент (get_Item)
Возвращает элемент по указанному индексу.
Синтаксис: Элемент(<Индекс>)

Имя параметра	Описание
Индекс	Индекс элемента в вписке, от 0 до (Количество - 1).

Добавить (Add)
Добавляет в список новый элемент.
Синтаксис: Добавить(<Элемент>)

Удалить (Remove)
Удаляет из списка указанный элемент.
Синтаксис: Удалить(<Элемент>)

УдалитьПоИндексу (RemoveAt)
Удаляет из списка элемент по указанному индексу.
Синтаксис: УдалитьПоИндексу(<Индекс>)

Имя параметра	Описание
Индекс	Индекс элемента в вписке, от 0 до (Количество - 1).

Добавить (Add)
Добавляет в коллекцию метку с указанным Tag ID.
Синтаксис: Добавить (<tagId>)

Имя параметра	Описание
tagId	Tag ID метки в виде строки в 16-ричном формате.

Cleverence.RFID.RfidTag.EPCUII_BANK

Отражает содержимое банка 01 (EPC) RFID-меток типа Class 1 Generation 2.

IsValid (IsValid)
Возвращает true, если данные банка памяти корректны и соответствуют стандарту.

ErrorString (ErrorString)
Возвращает описание ошибки для некорректных данных банка памяти.

ОригинальныеДанные (OriginalData)
Для банка, прочитанного из метки, возвращает оригинальные бинарные данные. Для экземпляра, созданного пользователем, возвращает "Неопределено".

БинарноеПредставление (BinaryString)
Возвращает строку 16-ричного представления сожержимого данного банка.

EPCUII (EPCUII)
Возвращает EPC (Electronic Product Code http://www.gs1.org/aboutepc/essential), прошитый в банке 01, или UII (Unique Item Identifier), если метка в соответствии со стандартами ISO.

СоответствуетEPCglobal (IsEPGglobalCompliant)
Возвращает true (Истина), если метка закодирована в соответствии со стандартом EPCglobal.

СоответствуетISO15961 (IsISO15961Compliant)
Возвращает true (Истина), если метка закодирована в соответствии со стандартом ISO 15961.

CRC16 (CRC16)
Возвращает чексумму CRC16 всех данных банка.

PC (PC)
Возвращает PC-часть (Protorol Control) заголовка банка 01.

Cleverence.RFID.RfidTag.TID_BANK

Отражает содержимое банка 10 (TID) RFID-меток типа Class 1 Generation 2.

ОригинальныеДанные (OriginalData)
Для банка, прочитанного из метки, возвращает оригинальные бинарные данные. Для экземпляра, созданного пользователем, возвращает "Неопределено".

СоответствуетEPCglobal (IsEPGglobalCompliant)
Возвращает признак того, что TID соответствует стандарту GS1 (см. GS1 Tag Data Standard (TDS) v 1.5, http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_5-standard-20100818.pdf).

MDID (MDID)
Возвращает или устанавливает международный код производителя чипа RFID-метки (Tag Mask Designer Identifier), максимум 12 бит.

TMN (TMN)
Возвращает или устанавливает номер модели чипа RFID-метки (Tag Model Number, согласно внутреннему каталогу производителя чипа), максимум 12 бит.

МодельЧипа (ChipModel)
Возвращает модель чипа, если она известна компоненте.

СерийныйНомер (SerialNumber)
Возвращает серийный номер чипа.

БинарноеПредставление (BinaryString)
Возвращает строку 16-ричного представления сожержимого данного банка.

Uri (Uri)
Возвращает STID URI согласно стандарта GS1.

Cleverence.RFID.RfidTag.USER_BANK

Отражает содержимое банка 11 (USER) RFID-меток типа Class 1 Generation 2.

Cleverence.RFID.RfidTag.RESERVED_BANK

Отражает содержимое банка 00 (RESERVED) RFID-меток типа Class 1 Generation 2.

ОригинальныеДанные (OriginalData)
Для банка, прочитанного из метки, возвращает оригинальные бинарные данные. Для экземпляра, созданного пользователем, возвращает "Неопределено".

ПарольДоступа (AccessPassword)
Возвращает или устанавливает пароль на доступ к меткам. Число "0" означает отсутствие пароля.

ПарольНаБлокирование (KillPassword)
Возвращает или устанавливает пароль на операцию "убийства" меток. Число "0" означает отсутствие пароля.

ДополнительныеБайты (ExtededBytes)
Возвращает или устанавливает дополнительные расширенные байты банка 02 (есть они, нет, и сколько их - зависит от конкретного чипа метки).

БинарноеПредставление (BinaryString)
Возвращает строку 16-ричного представления сожержимого данного банка.

Cleverence.RFID.GS1.Epc

Реализация стандарта GS1 на электронный код товара EPC (Electronic Product Code http://www.gs1.org/aboutepc/essential). (см. GS1 Tag Data Standard (TDS) v 1.5, http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_5-standard-20100818.pdf).

Схема (EpcScheme)
Возвращает вариант схемы, которой соответствует данный EPC.
(пока поддерживаются только схемы кодирования SGTIN-96, SSCC-96, GRAI-96, GIAI-96 и GDTI-96).

AttributeBits (AttributeBits)
Возвращает дополнительные флаги (см. «AttributeBits (AttributeBits)»)

КодКомпании (Company)

Возвращает код компании, зарегистрированной в GS1.

Ссылка (Reference)
Возвращает ссылку на объект в каталоге. В зависимости от схемы кодирования EPC эта ссылка будет означать либо код товара согласно каталога компании, либо тип документа, либо код вида возвратной тары, либо тип места назначения и т.п.

СерийныйНомер (Serial)
Возвращает серийный номер конкретного объекта.

Фильтр (FilterValue)
Возвращает значение Filter Value из бинарного кодирования.
URI (URI)
Возвращает EPC URI (EPC pure identity URI) согласно стандарта GS1.

SGTIN (SGTIN)
Возвращает объект, содержащий поля SGTIN, либо "Неопределено", если данный EPC кодирует не SGTIN (т.е., не товар, не аксессуар и не упаковка для товаров).

GDTI (GDTI)
Возвращает объект, содержащий поля GDTI, либо "Неопределено", если данный EPC кодирует не GDTI (т.е. это не документ и не контейнер для документов)

SSCC (SSCC)
Возвращает объект, содержащий поля SSCC, либо "Неопределено", если данный EPC кодирует не SSCC (т.е. это не палета и не контейнер).

Строка (ToString)
Возвращает строкое представление данного EPC (URI либо описание ошибки).
Синтаксис: Строка(), метод не принимает аргументов.

Cleverence.RFID.ISO.Uii

Представляет собой уникальный идентификатор объекта (UII, Unique Item Identifier), в рамках определенной сферы применения, согласно стандарту ISO 15961 (Radio frequency identification (RFID) for item management - Data protocol: application interface).

AFI (AFI)
Возвращает значение кода применения (см. «Afi (Afi)»)

Value (Value)
Возвращает синтетический уникальный код, сформированный на основе данных UII.

Cleverence.RFID.ISO.LibraryUii

Представляет собой (UII, Unique Item Identifier, уникальный идентификатор) для библиотечного объекта согласно стандарту ISO 28560.

ItemId (ItemId)
Возвращает или устанавливает номер библиотечного объекта (Primary Item Identifier), уникальный в рамках конкретной библиотеки.

ISIL (ISIL)
Возвращает или устанавливает ISIL библиотеки-владельца объекта. Необязательный параметр. Строку ISIL можно хранить в USER-банке RFID-метки. Национальным агентством по присвоению кодов ISIL в России является ГПНТБ. Строка ISIL должна соответствовать ISO 15511.

ТипИспользования (TypeOfUsage)
Возвращает или устанавливает тип использования объекта. Тип использования определяет, что это: объект фонда, читательский билет или собственное имущество библиотеки (стол, принтер).

Закодировать (Encode)
Выполняет кодирование UII библиотечного объекта в байты в соответствии со стандартом ISO 28560.
Синтаксис: Закодировать (<alignToWords>)
Возвращает: Массив байт бинарно закодированного UII или пустой массив, если UII некорректен (см. «IsValid (IsValid)»).

Cleverence.RFID.ISO.LibraryItem

Содержит записи о библиотечном объекте, пригодные для записи в USER банк RFID-метки согласно стандарту ISO 28560.

УникальныйКод (PrimaryItemIdentifier)
Возвращает или устанавливает строковой код библиотечного объекта, уникальный в рамках конкретной библиотеки. Устанавливать не обязательно, т.к. этот же самый код уже должен храниться в банке EPC/UII используемой RFID-метки.

ISIL (ISIL)
Возвращает или устанавливает ISIL библиотеки, которой принадлежит объект. Национальным агентством по присвоению кодов ISIL в России является ГПНТБ. Строка ISIL должна соответствовать ISO 15511.

Наименование (Title)
Возвращает или устанавливает заголовок (название) библиотечного объекта. Можно использовать все символы Юникод.

МестоНаПолке (ShelfLocation)
Возвращает или устанавливает номер полки для хранения библиотечного объекта (строка).

РазмерНабора (SetSize)
Возвращает или устанавливает количество объектов в библиотечном наборе (например, общее число томов в издании). Если это не набор или размер набора неизвестен, то равно нолю.

ИндексВНаборе (SetIndex)
Возвращает или устанавливает номер объекта в библиотечном наборе (например, номер тома). Если это не набор, то равно нулю.
Если это первый элемент набора и при этом известно, что в наборе присутствуют объекты без RFID-меток, то тоже должно быть равно нулю (соотв. у следующего объекта в наборе номер должен быть не 1, а сразу 2).

СформироватьUSERБанк (ToUSER_BANK)
Создает и заполняет экземпляр USER-банка памяти RFID-метки на основе данного набор записей о библиотечном объекте.
Синтаксис: СформироватьUSERБанк(), метод не принимает аргументов.
Возвращает: Созданный и заполненный в соответствии с ISO 28560 USER-банк памяти RFID-метки.

Cleverence.RFID.ANSI.AnsiUii

Вариант UII по стандарту ANSI, поля которого регламентируются стандартом MH 10.8.2.

Элемент (Item)
Возвращает или устанавливает строковое значение поля по указанной строке Data Identifier.

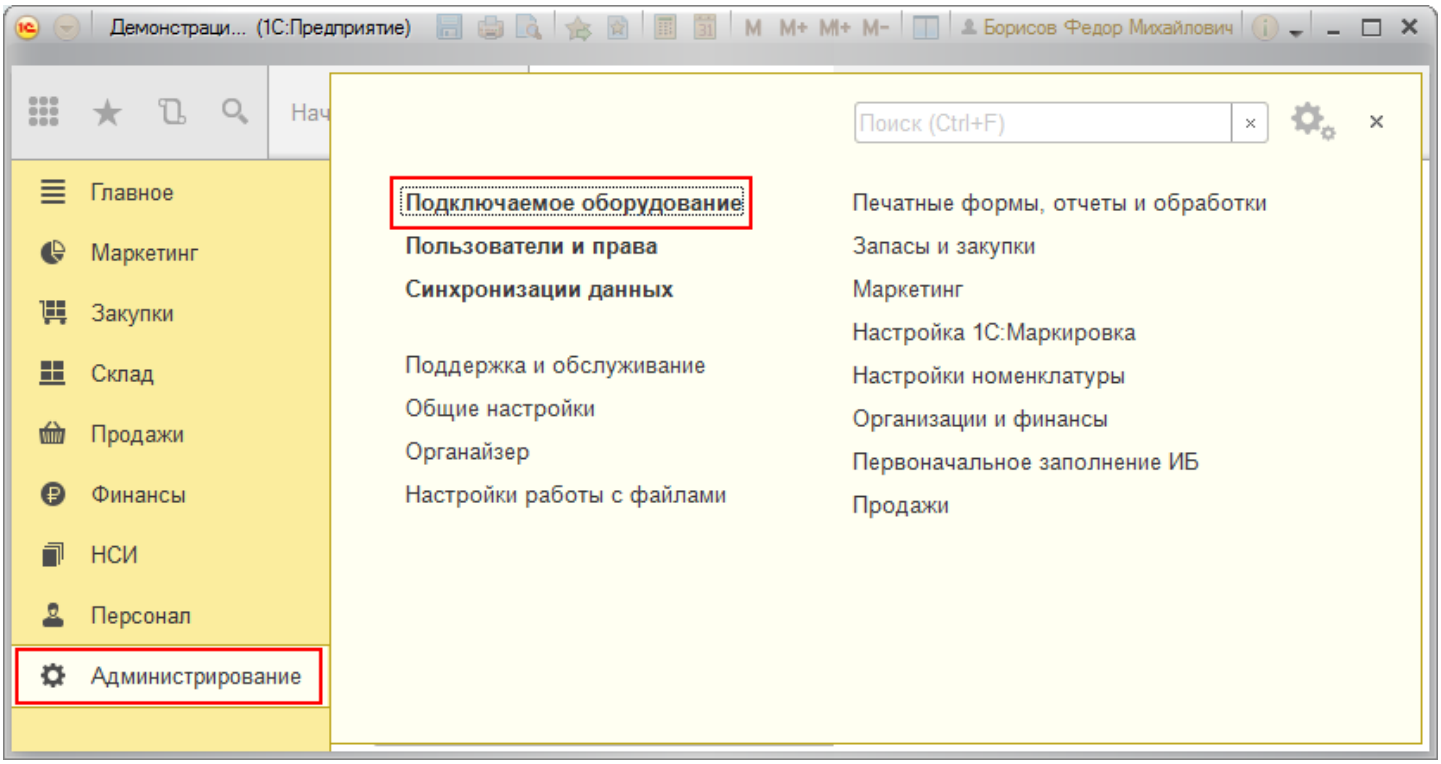
Задать вопрос в техническую поддержку

Подключение RFID-компоненты в 64-битной версии «1С: Предприятия»

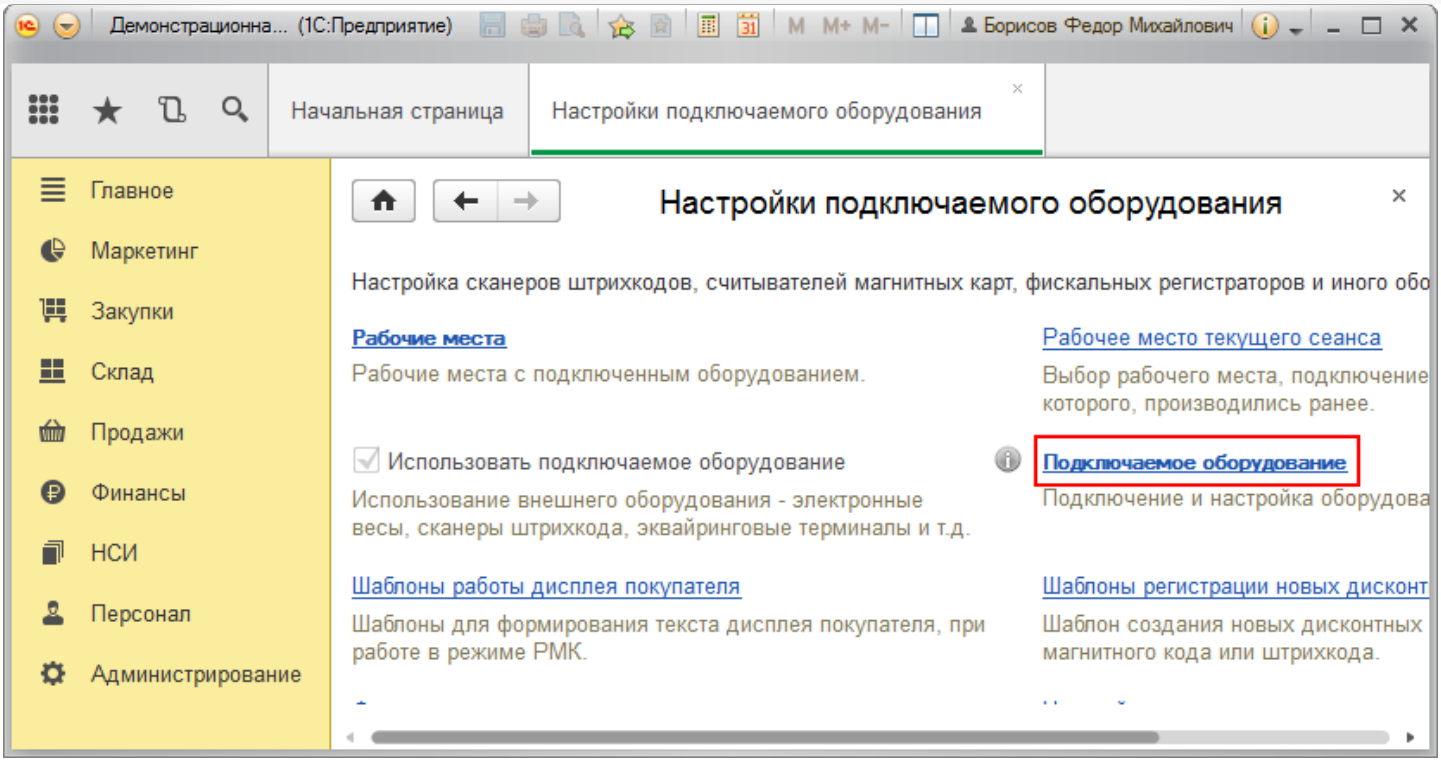
Последние изменения: 2024-03-26

> **Только для тех, у кого установлена 64-битная версия 1С. Для 32-битной версии в 1С уже есть встроенная компонента.**

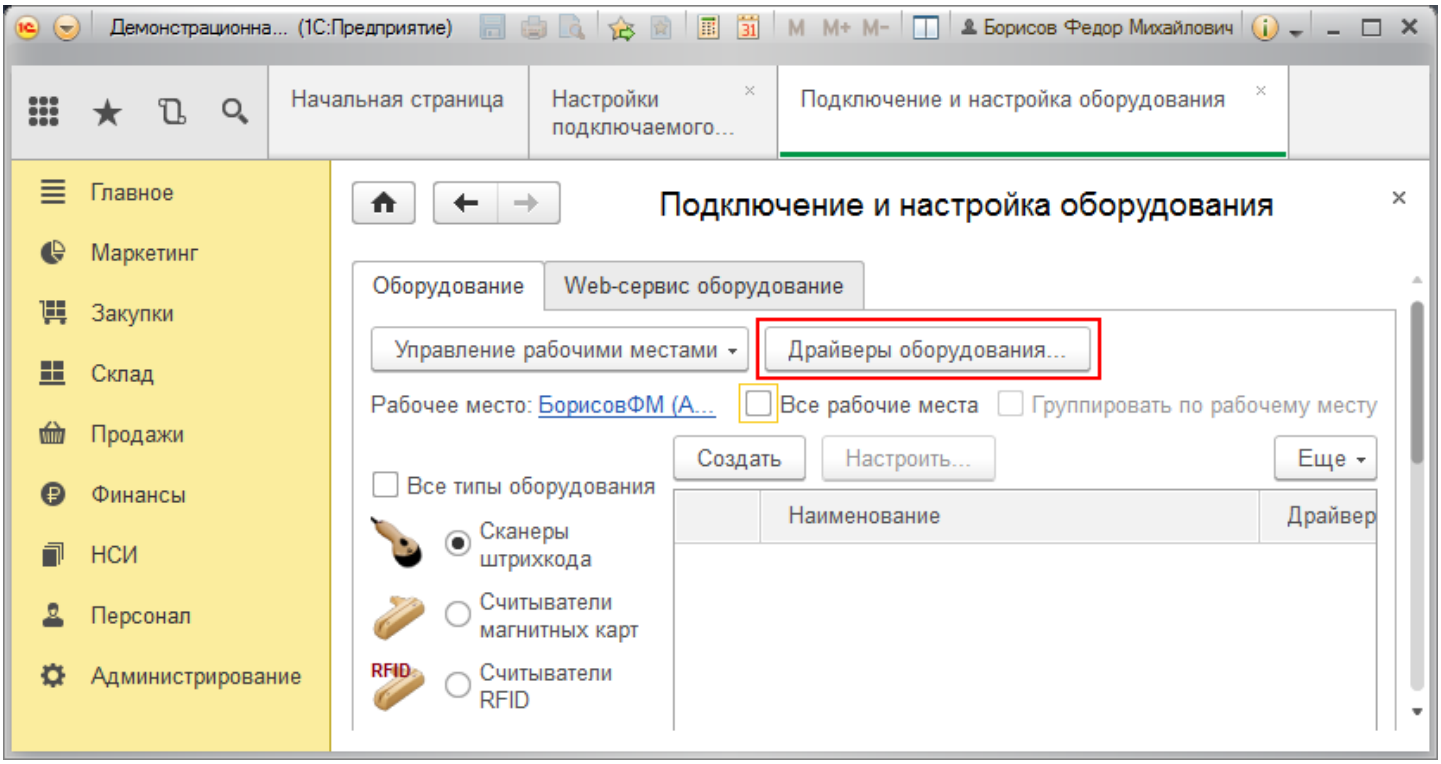
Для подключения RFID-компоненты в 64-битной версии «1С: Предприятия» зайдите в раздел «Администрирование» --> «Подключаемое оборудование».



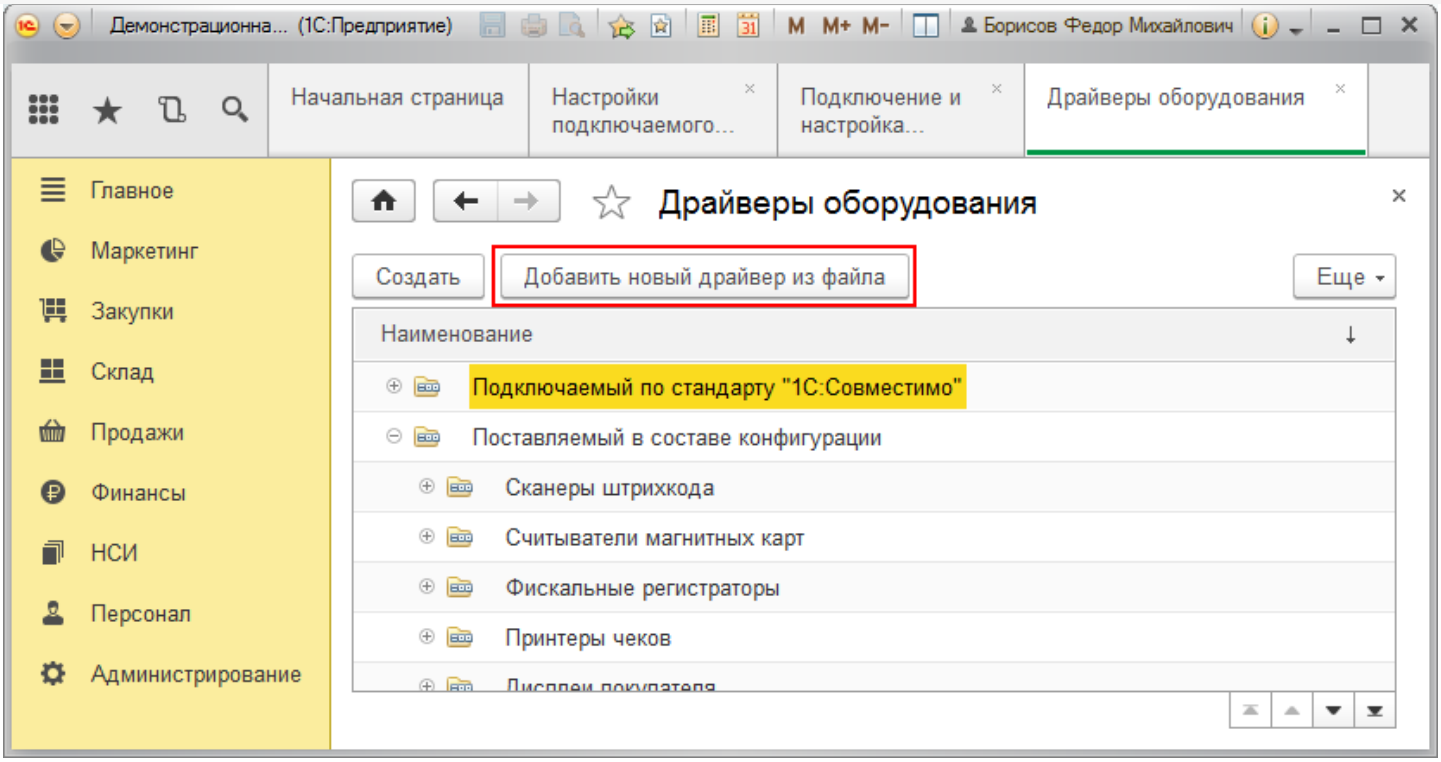
Откройте вкладку «Подключаемое оборудование».



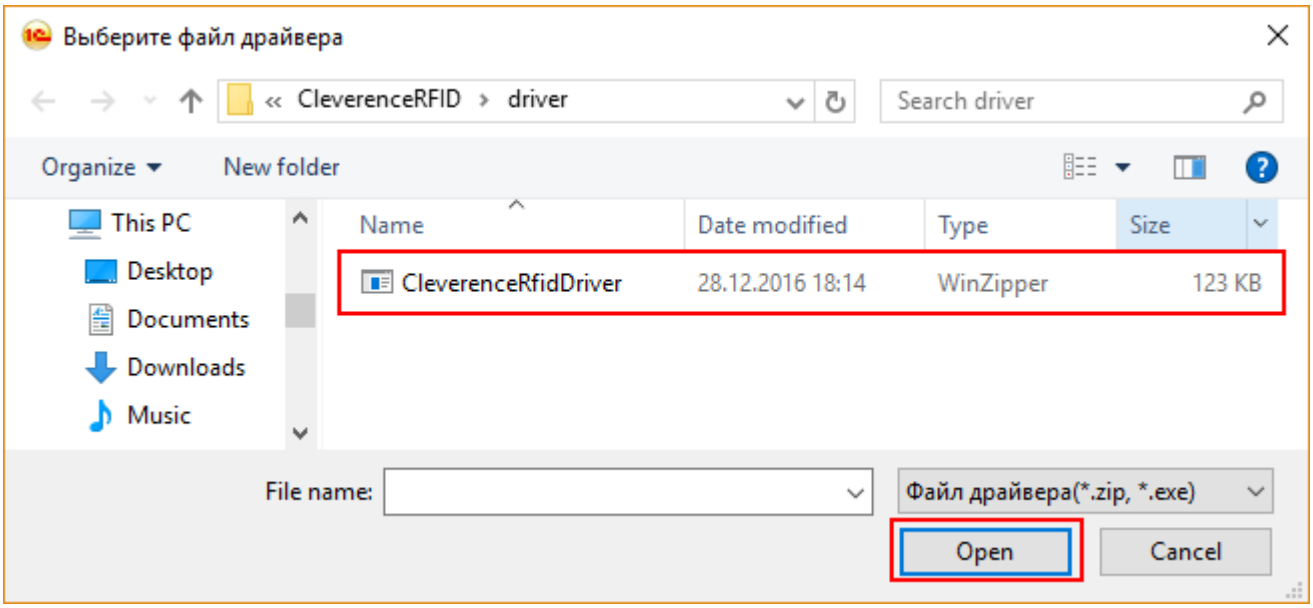
Нажмите кнопку «Драйверы оборудования».



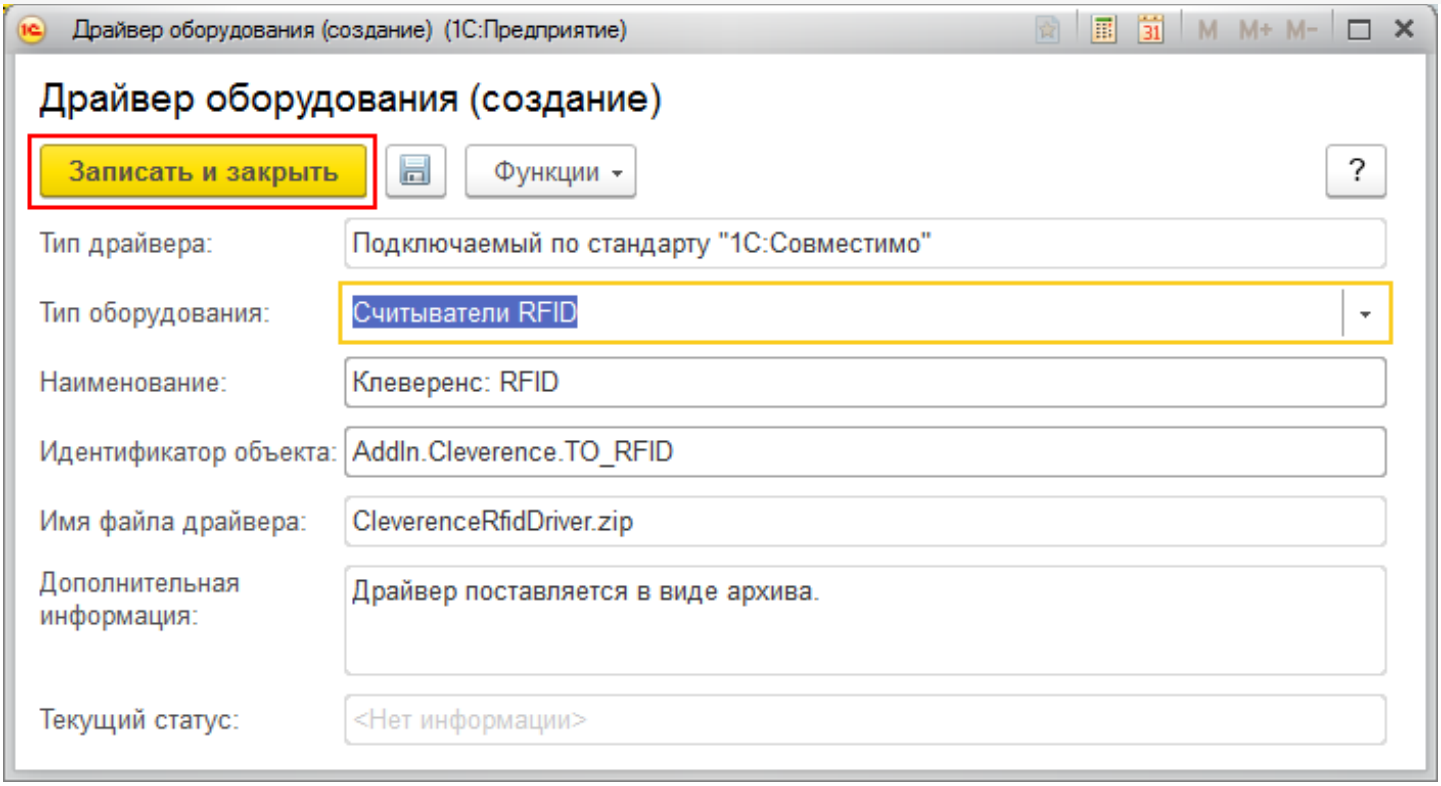
Добавьте новый драйвер из файла.



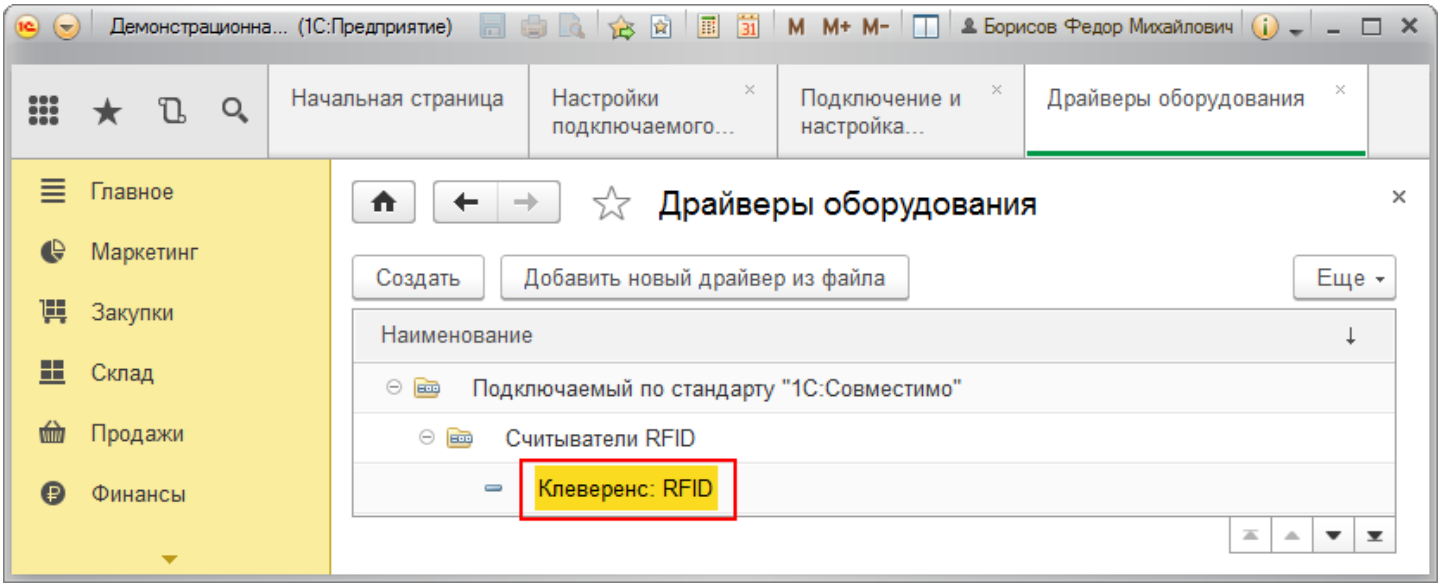
Выберите файл «CleverenceRfidDriver.zip» (по умолчанию путь к файлу «C:\Program Files (x86)\Cleverence Soft\CleverenceRFID\driver\CleverenceRfidDriver»).



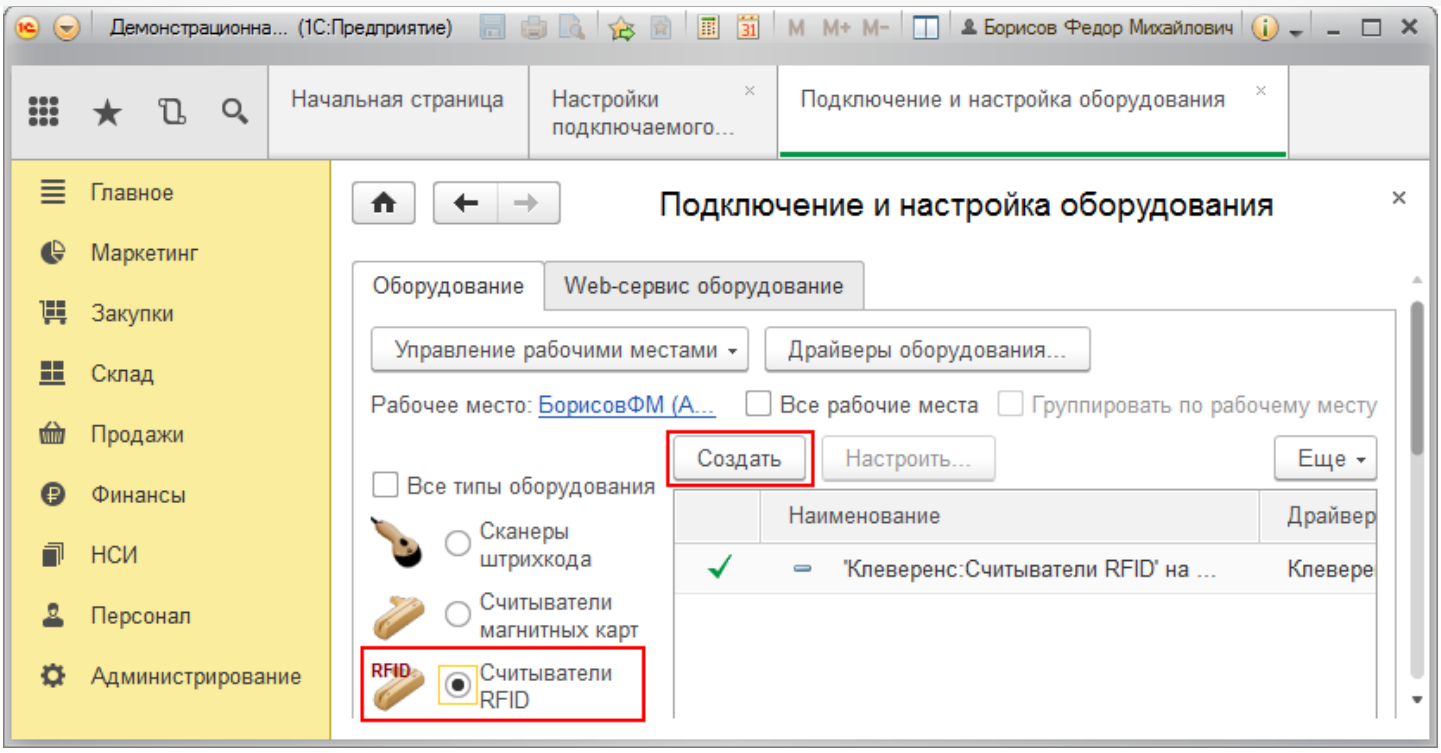
Запишите и закройте созданный драйвер оборудования.



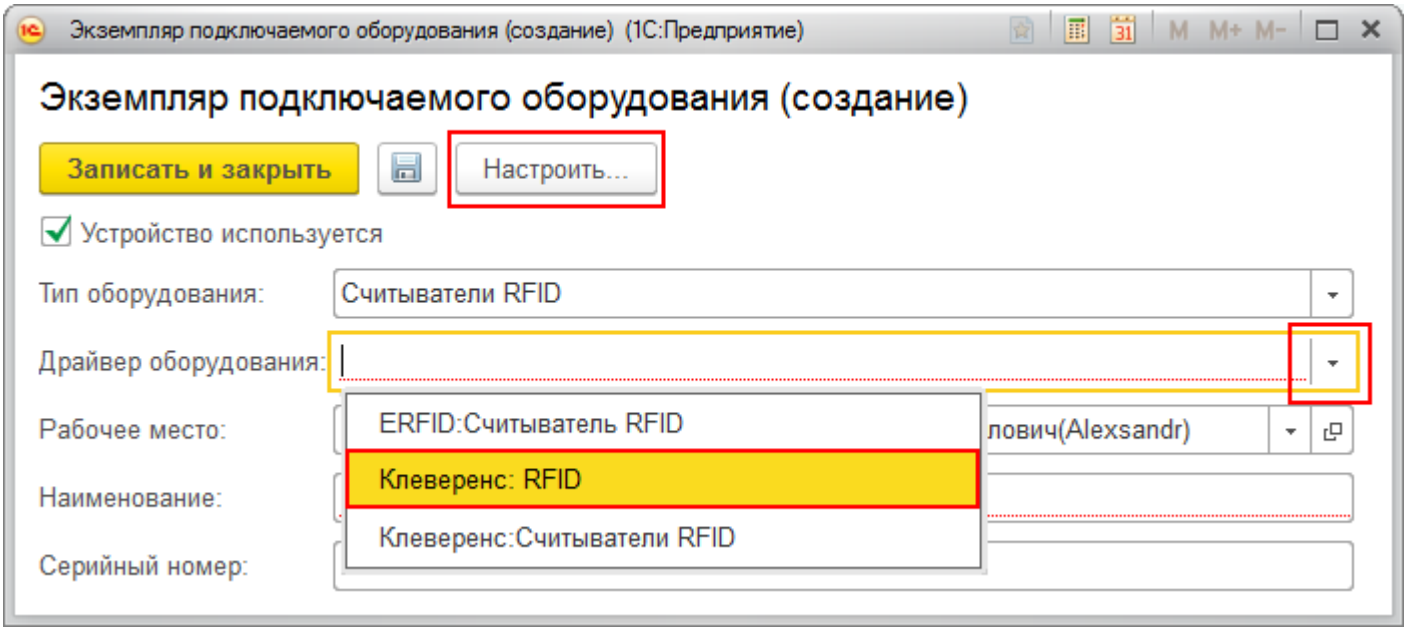
Драйвер добавлен.



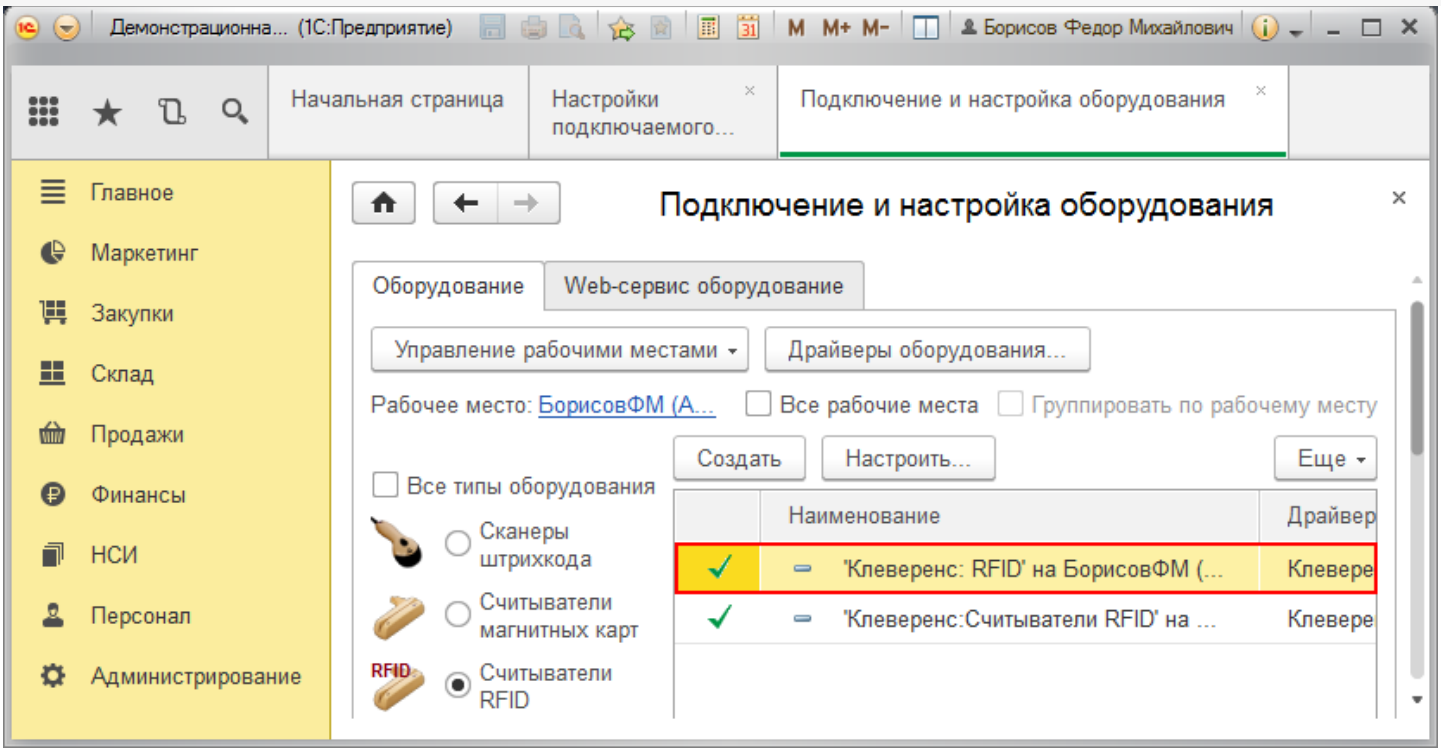
Теперь необходимо подключить оборудование.



Выберите добавленный вами драйвер и настройте его.



Оборудование подключено, настроено и готово к использованию.



интеграция, 1С, RFID

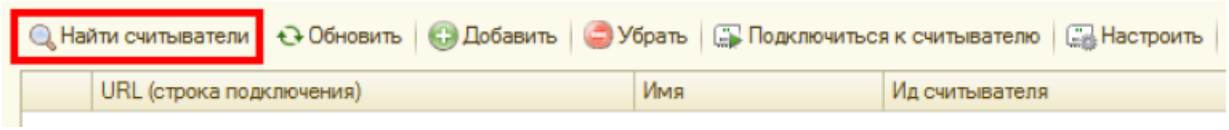
Не нашли что искали?

Задать вопрос в техническую поддержку

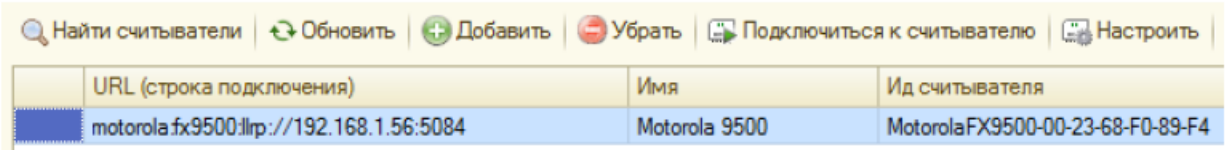
Поиск и подключение RFID-считывателей с помощью демонстрационной обработки 1С

Последние изменения: 2024-03-26

Если ваш RFID-считыватель находится в той же локальной сети, что и компьютер, на котором запущен клиент 1С, то вы можете воспользоваться поиском считывателей:



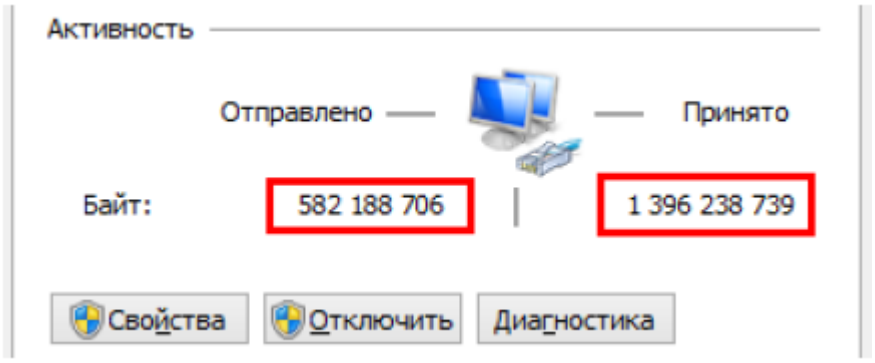
На время поиска (порядка 30 секунд) 1С «зависает» (не отвечает). Результат поиска отображается ниже в таблице считывателей:



Компонента позволяет найти все поддерживаемые считыватели в локальной подсети.

Если считыватель не находится:

1. Прочтите статью об установке и настройке вашего считывателя.
2. Проверьте, что считыватель включен, кабели подключены, все лампочки зеленые.
3. Если считыватель подключен к ПК по кабелю USB, зайдите в «Панель управления» --> «Сеть и Интернет»--> «Центр управления сетями и общим доступом»--> «Изменение параметров адаптера» и убедитесь, что виртуальный адаптер RNDIS подключен, на иконке нет красного креста, в окне состояния (двойной клик на иконке) показано, что он активен и данные ходят туда-сюда:



4. Если считыватель подключен по кабелю Ethernet:
   - убедитесь, что кабель воткнут в сеть и огоньки под кабелем горят зеленым:
   - убедитесь, что считыватель может получить доступ в вашу локальную сеть и получить IP
   - убедитесь, что считыватель в сети не блокируется сетевым коммутатором, файерволом, антивирусом и т.п.

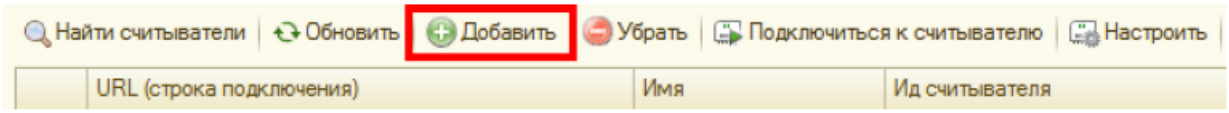
Если ничего не помогает:

1. Найдите свободный компьютер или ноутбук с сетевой картой (разъемом для кабеля Ethernet), который можно было бы временно отключить от общей сети. Выключите на нём все сетевые адаптеры, кроме того, который отвечает за кабель Ethernet:

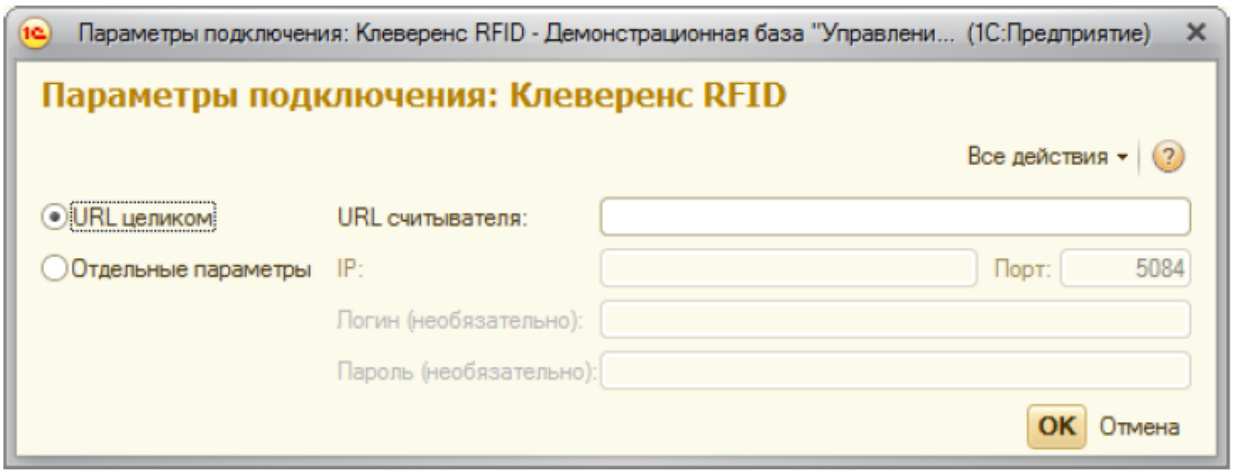
2. Установите на этом компьютере IP 192.168.0.1, включите DHCP.
3. Найдите правильно обжатый Ethernet кабель (компьютер-компьютер) и подключите им считыватель к компьютеру напрямую. Среди выданных DHCP IP вы должны будете видеть выданный считывателю IP.

Подключение вручную

Если считыватель не находится автоматически, но вы точно знаете, что он есть в сети и знаете его IP, то вы можете добавить считыватель вручную:



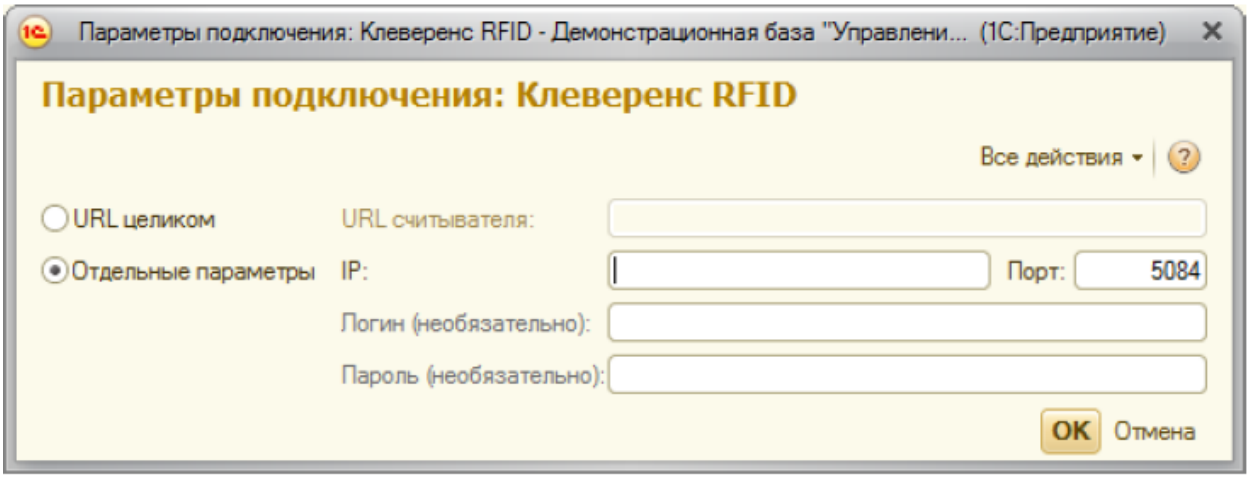
По URL, который имеет специальный формат и может содержать в себе все параметры подключения:



Примеры URL:

- 10.10.0.121
- http://10.10.0.121
- motorola:llrp://10.10.0.121
- motorola:llrp://10.10.0.121:5084
- motorola:fx7400:llrp://10.10.0.121:5084
- motorola:fx9500:llrp://10.10.0.121:5084
- motorola:fx9500:llrp://10.10.0.121
- motorola:fx9500:llrp://10.10.0.121 login=admin, password=change
- motorola:llrp://10.10.0.121 login=admin, password=change

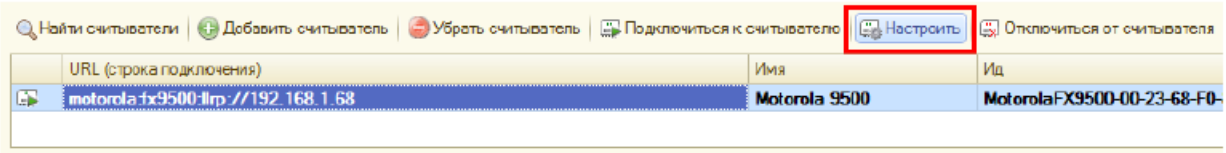
Либо по IP и другим отдельным параметрам подключения.

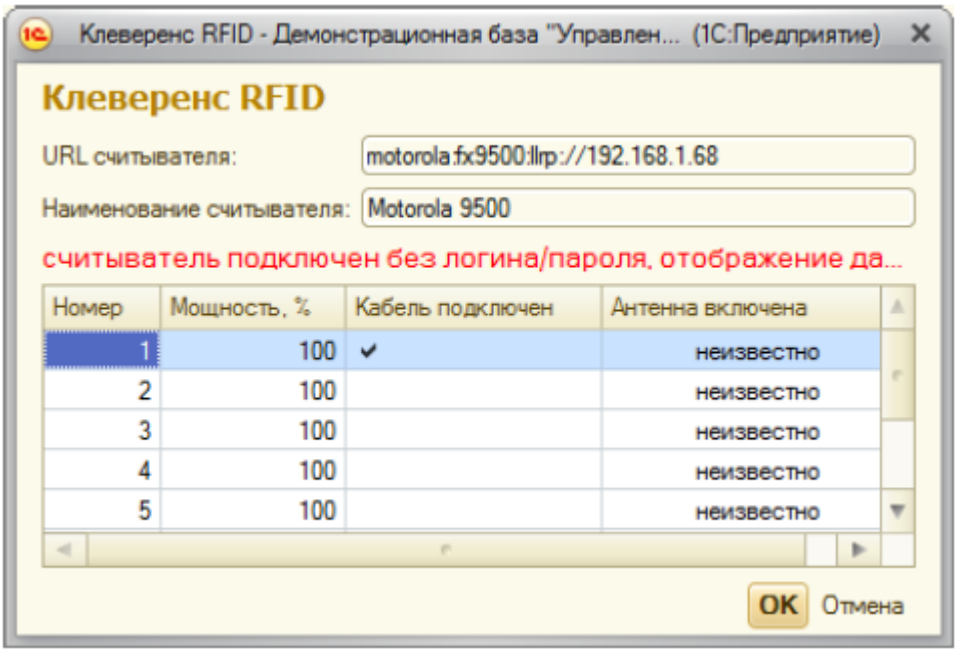


Логин и пароль могут понадобиться для управления антеннами. Стандартные логины и пароли на ваш считыватель ищите в статье «Установка и настройка RFID считывателей»

Настройка считывателей

Найденные или добавленные считыватели можно настроить при помощи окна настройки считывателя:

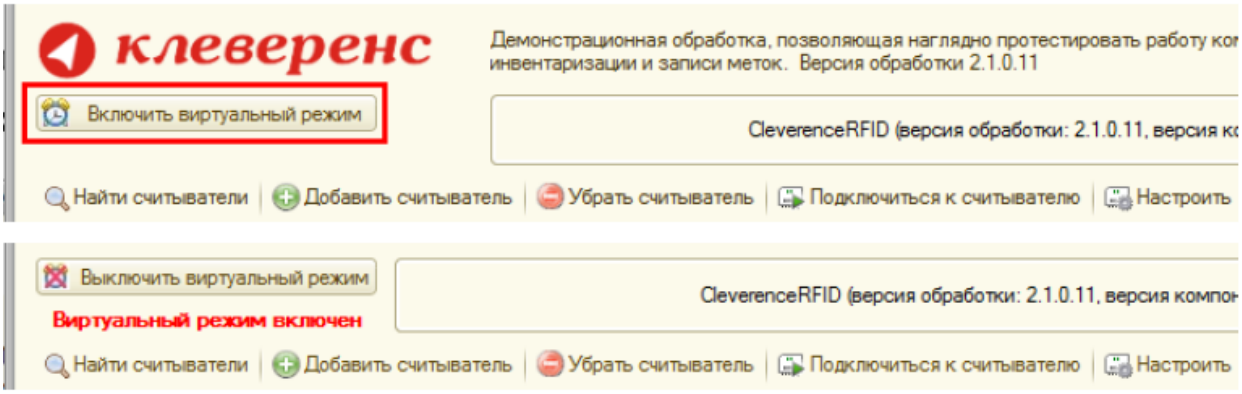




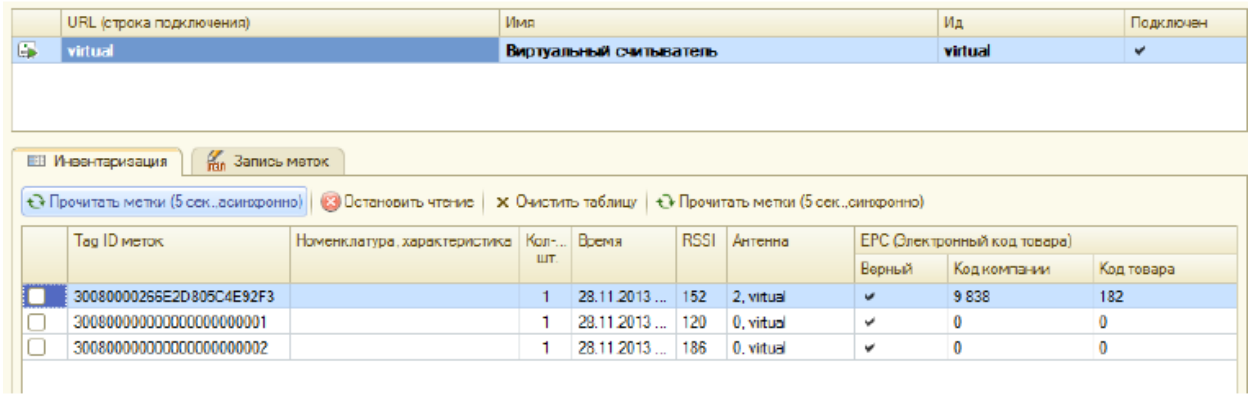
Из доступного – изменение URL и управление мощностью подключенных антенн.

Тестирование компоненты без RFID-считывателей

Для тестирования работы компоненты без RFID-считывателя на руках, в ней предусмотрен так называемый «виртуальный режим», в котором компонента подключается к виртуальным считывателям и читает виртуальные метки. Чтобы протестировать работу компоненты без считывателей, в демонстрационной обработке предусмотрена кнопка «Включить виртуальный режим»:

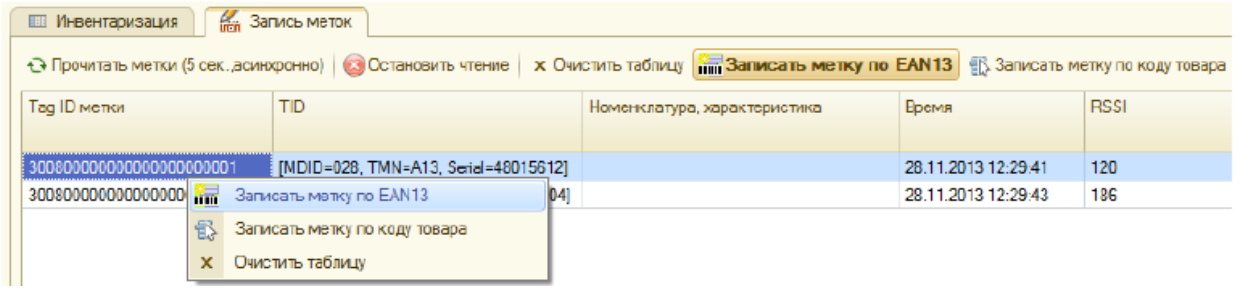


В виртуальном режиме все RFID-считыватели в окне демообработки являются фиктивными и компонента на самом деле ни к одному из них не подключается. Все метки, которые будут якобы читаться компонентой (пока она находится в виртуальном режиме) тоже фиктивные:



В стандартной настройке демонстрационной обработки виртуальная инвентаризация читает 2 (две) конкретные метки всегда + иногда еще 0-2 случайные метки.

В фиктивные метки, прочитанные в виртуальном режиме, даже можно писать (только в «Управлении торговлей 11»):



При этом писаться будут только те две конкретные метки, а в дополнительные случайные метки писать не имеет смысла, т.к. они генерируются случайно и читаются только один раз.

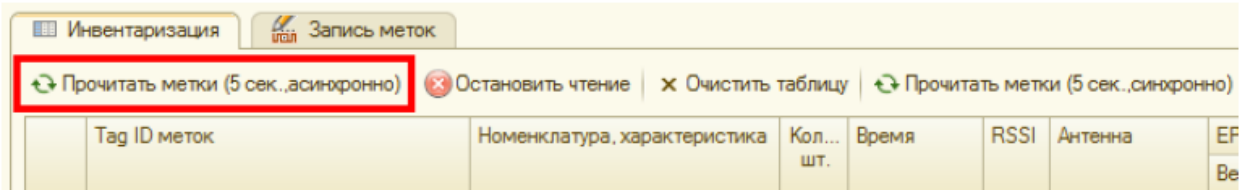
Не нашли что искали?

Задать вопрос в техническую поддержку

Инвентаризация RFID-меток с помощью демонстрационной обработки 1С

Последние изменения: 2024-03-26

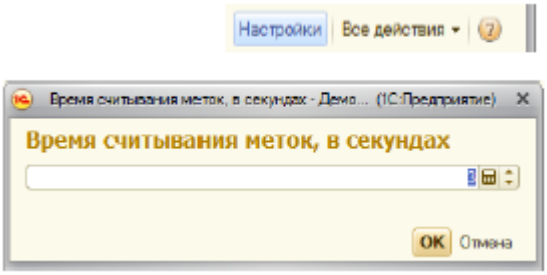
Демонстрационная обработка позволяет провести инвентаризацию RFID-меток, находящихся в поле зрения антенн подключенных RFID-считывателей.



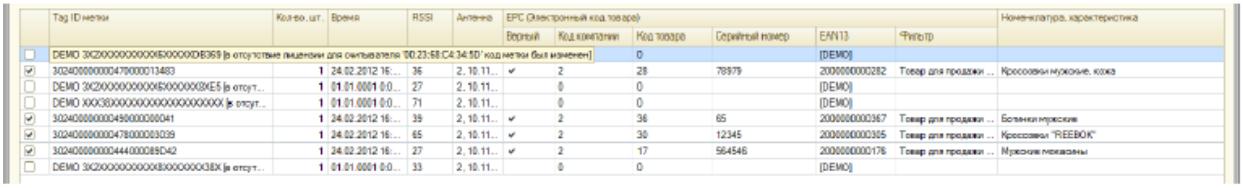
При синхронном чтении «1С:Предприятие» замирает на время чтения, после чего считанные метки появляются сразу все.

При асинхронном чтении «1С:Предприятие» не замирает, а метки появляются одна за другой по мере их считывания. Это наиболее удобный и интересный режим, который доступен не всегда.

Время чтения метки можно задать в настройках обработки:



Считанные метки отображаются в таблице, их EPC декодируются:



> В отсутствие лицензии продукта для конкретного считывателя, которым прочитана метка, Tag ID метки может быть изменен на строку «DEMO ...» и текст об отсутствии лицензии.

В приведенной таблице колонка «Количество» отображает сколько меток с идентичной номенклатурой (или одинаковыми Tag ID) было прочитано.

Колонка «RSSI» – условный уровень сигнала от метки по шкале от 0 до 100.

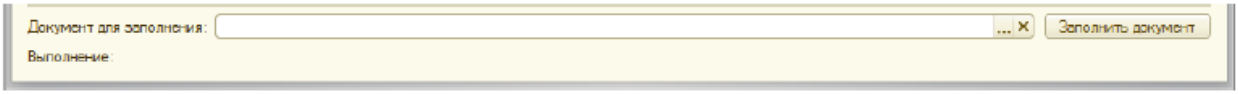
В колонке «Антенна» показан номер антенны считывателя (от 1 до ....) и URL самого считывателя (если читать сразу с нескольких считывателей).

В колонке «EAN13» показан синтетический штрихкод, сгенерированный по данным EPC метки.

Колонка «Номенклатура» видна только в том случае, если в конфигурации (в которой открыта демонстрационная обработка) присутствует справочник номенклатуры. Товары ищутся либо по коду товара из

EPC, либо по синтетическому штрихкоду EAN13.

Если демонстрационная обработка открыта в конфигурации «Управлении торговлей 11», то в нижней части окна обработки отображается раздел, позволяющий использовать данные о считанных метках для заполнения таблицы товаров какого-нибудь документа 1С:



В документ будут переноситься те строки из таблицы считанных меток, которые отмечены галочкой. По умолчанию обработка отмечает галочкой все метки, по которым было найдено соответствие какому-либо товару базы «1С:Предприятия».

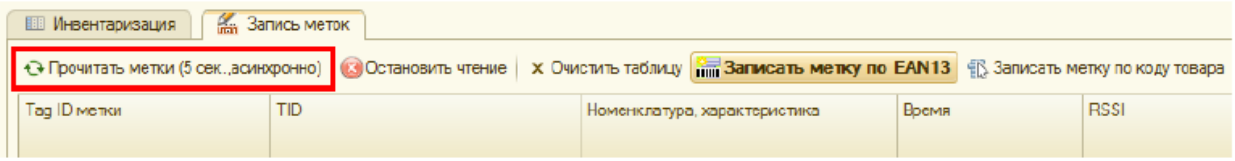
Не нашли что искали?

Задать вопрос в техническую поддержку

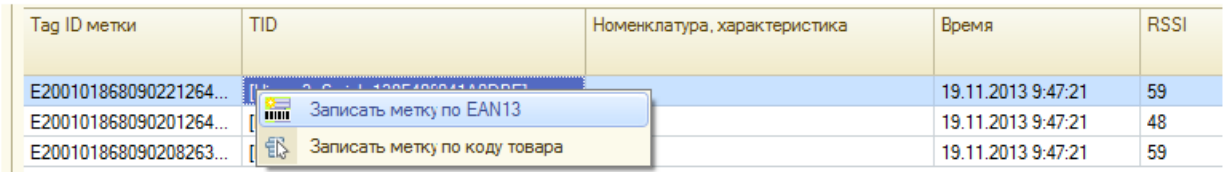
Запись RFID-меток с помощью демонстрационной обработки 1С

Последние изменения: 2024-03-26

Если в конфигурации (в которой открыта демонстрационная обработка) присутствует справочник номенклатуры, демонстрационная обработка позволяет записать в метки данные о товарах из базы «1С:Предприятия». Для этого необходимо переключиться на закладку «Запись меток» и прочитать метки:



Затем можно выбрать любую из прочитанных меток и записать в неё информацию о товаре:



В отсутствие лицензии на продукт для считывателя, которым прочитана интересующая метка, Tag ID метки может быть заменен на строку «DEMO ...» и текст об отсутствии лицензии. Метки с текстом DEMO недоступны для записи. Однако это не значит, что такую метку совсем нельзя записать – метки заменяются на DEMO в случайном порядке, поэтому можно попытаться еще раз нажать одну из кнопок «Прочитать метки...» до тех пор, пока интересующая метка не будет нормально прочитана.

Запись происходит упрощенно по следующему алгоритму:

1. Обработка генерирует EPC либо на основе штрихкода EAN13, либо по числовому коду 1С товара из базы «1С:Предприятия»;
2. Сгенерированный EPC записывается в банк 01 (EPC-bank) выбранной RFID-метки.

Сразу после записи обработка выполняет повторное чтение меток, чтобы можно было увидеть результат записи.

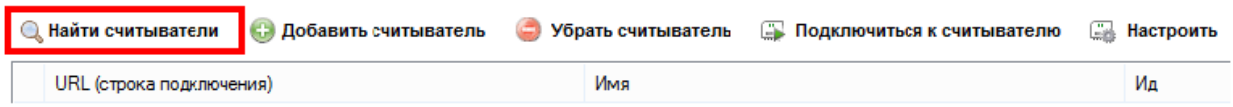
Не нашли что искали?

Задать вопрос в техническую поддержку

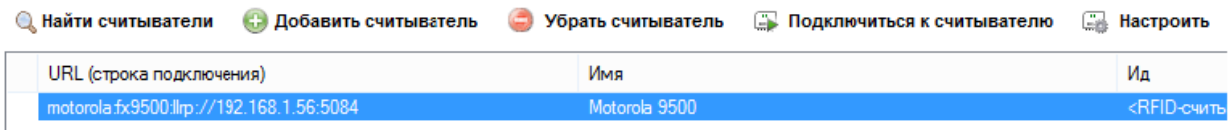
Поиск и подключение считывателей с помощью демо-программы

Последние изменения: 2024-03-26

Если ваш RFID-считыватель находится в той же локальной сети, что и ПК, на котором запущена демо-программа, то вы можете воспользоваться поиском считывателей:



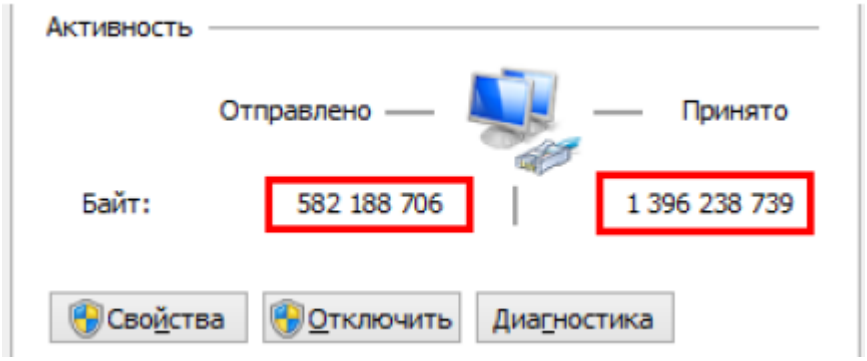
На время поиска (порядка 30 секунд) демо-программа «зависает» (не отвечает). Результат поиска отображается ниже в таблице считывателей:



Компонента позволяет найти все поддерживаемые считыватели в локальной подсети.

Если считыватель не находится:

1. Прочтите статью об установке и настройке вашего считывателя.
2. Проверьте, что считыватель включен, кабели подключены, все лампочки зеленые.
3. Если считыватель подключен к ПК по кабелю USB, зайдите в «Панель управления» --> «Сеть и Интернет»--> «Центр управления сетями и общим доступом»--> «Изменение параметров адаптера» и убедитесь, что виртуальный адаптер RNDIS подключен, на иконке нет красного креста, в окне состояния (двойной клик на иконке) показано, что он активен и данные ходят туда-сюда:



4. Если считыватель подключен по кабелю Ethernet:
   - убедитесь, что кабель воткнут в сеть и огоньки под кабелем горят зеленым:
   - убедитесь, что считыватель может получить доступ в вашу локальную сеть и получить IP
   - убедитесь, что считыватель в сети не блокируется сетевым коммутатором, файерволом, антивирусом и т.п.

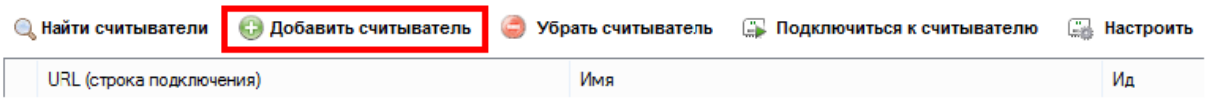
Если ничего не помогает:

1. Найдите свободный компьютер или ноутбук с сетевой картой (разъемом для кабеля Ethernet), который можно было бы временно отключить от общей сети. Выключите на нём все сетевые адаптеры, кроме того, который отвечает за кабель Ethernet:

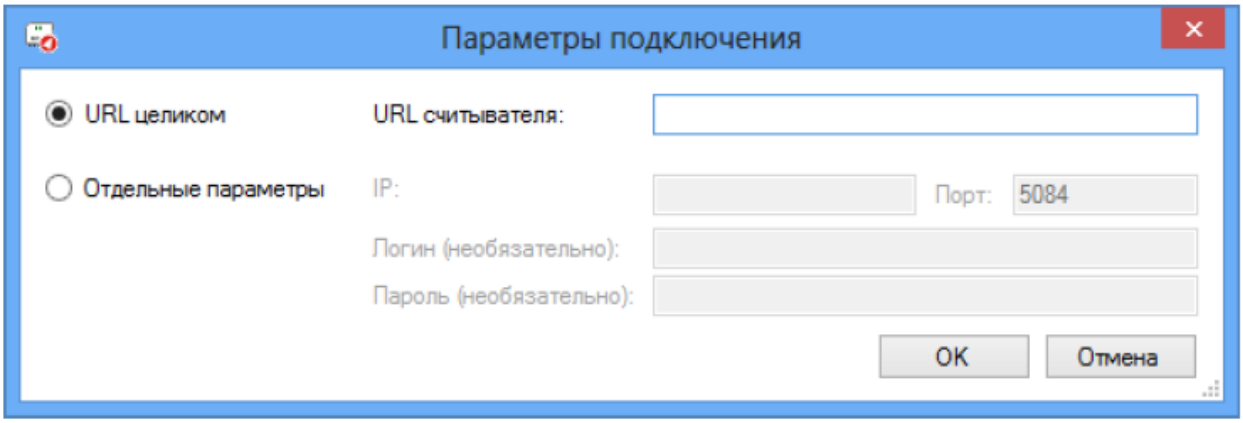
2. Установите на этом компьютере IP 192.168.0.1, включите DHCP.
3. Найдите правильно обжатый Ethernet кабель (компьютер-компьютер) и подключите им считыватель к компьютеру напрямую. Среди выданных DHCP IP вы должны будете видеть выданный считывателю IP.

Подключение вручную

Если считыватель не находится автоматически, но вы точно знаете, что он есть в сети и знаете его IP, то вы можете добавить считыватель вручную:



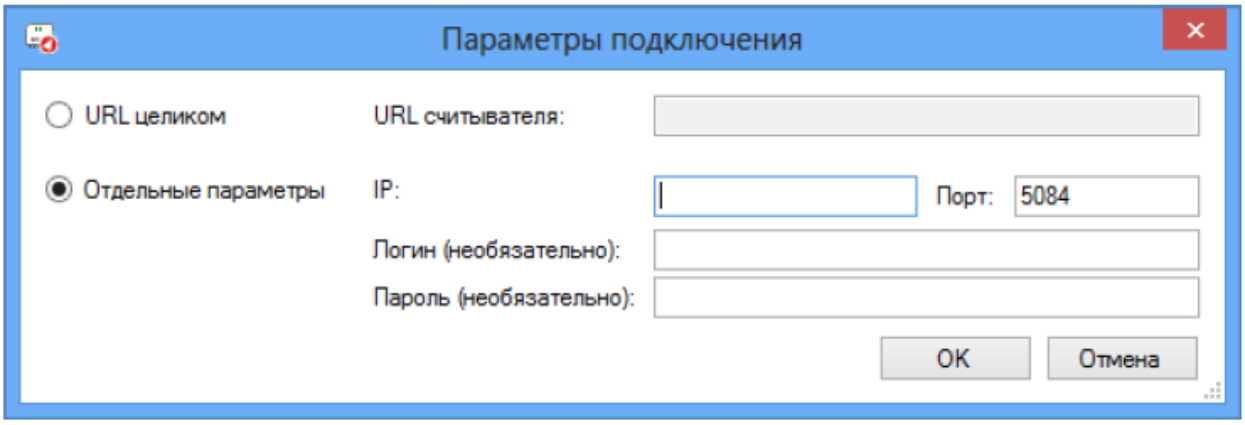
По URL, который имеет специальный формат и может содержатт в себе все параметры подключения:



Примеры URL:

- 10.10.0.121
- http://10.10.0.121
- motorola:llrp://10.10.0.121
- motorola:llrp://10.10.0.121:5084
- motorola:fx7400:llrp://10.10.0.121:5084
- motorola:fx9500:llrp://10.10.0.121:5084
- motorola:fx9500:llrp://10.10.0.121
- motorola:fx9500:llrp://10.10.0.121 login=admin, password=change
- motorola:llrp://10.10.0.121 login=admin, password=change

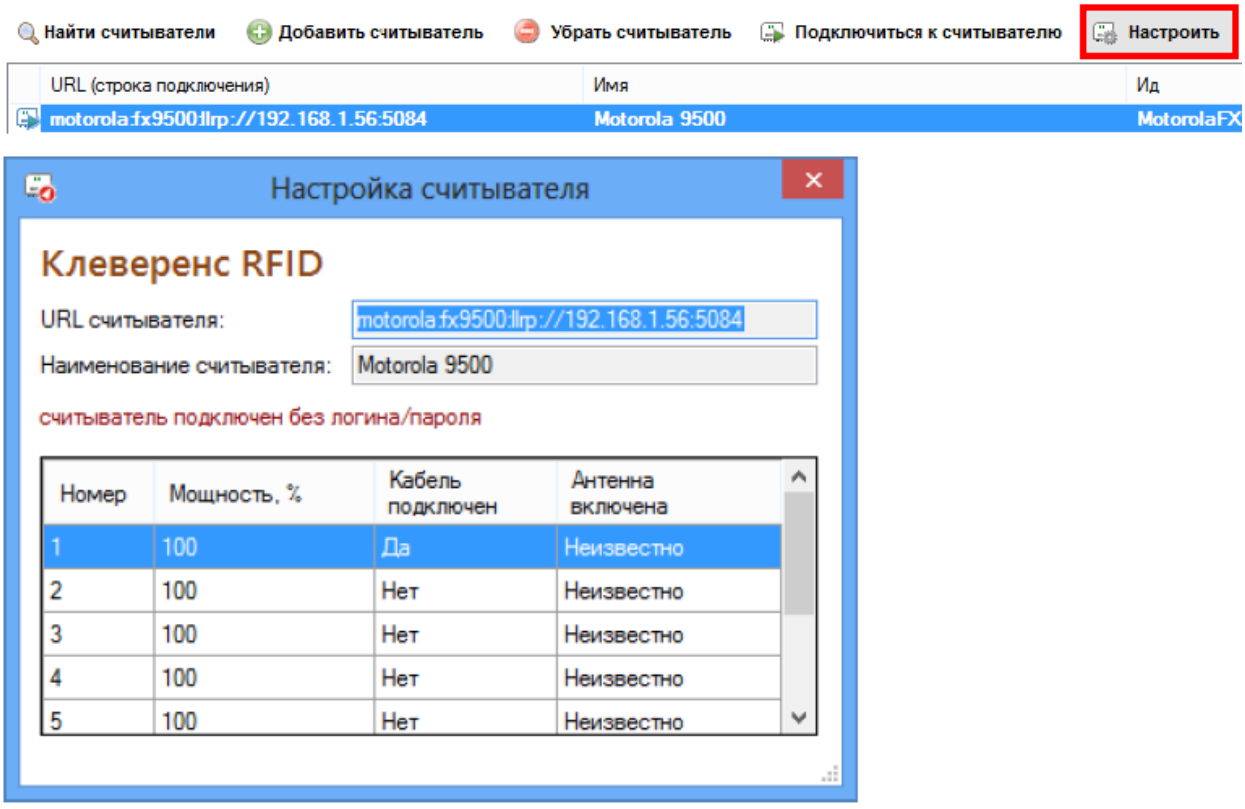
Либо по IP и другим отдельным параметрам подключения.



Логин и пароль могут понадобиться для управления антеннами. Стандартные логины и пароли на ваш считыватель ищите в статье «Установка и настройка RFID считывателей».

Настройка считывателей

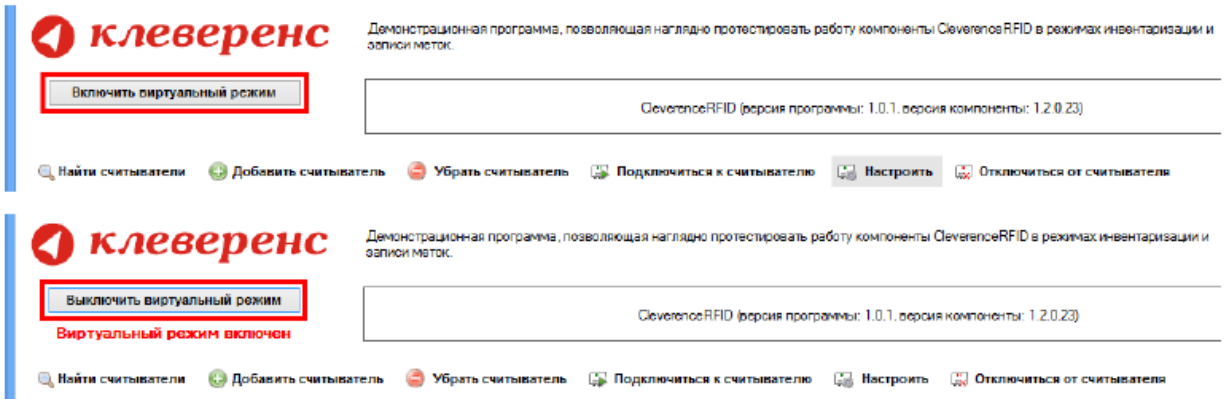
Найденные или добавленные считыватели можно настроить при помощи окна настройки считывателя:



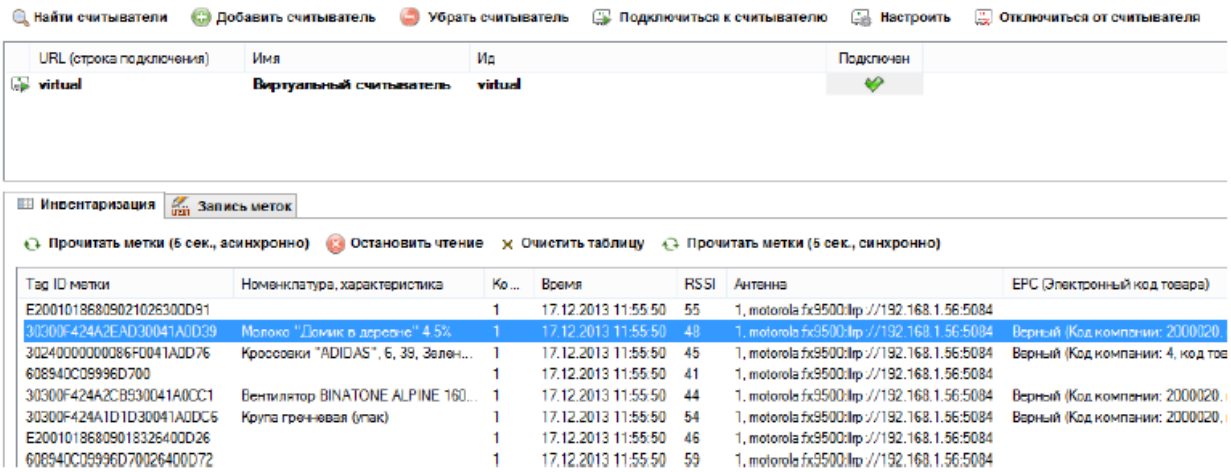
Из доступного – изменение URL и управление мощностью подключенных антенн.

Тестирование компоненты без RFID-считывателей

Для тестирования работы компоненты без RFID-считывателя на руках, в ней предусмотрен так называемый «виртуальный режим», в котором компонента подключается к виртуальным считывателям и читает виртуальные метки. Чтобы протестировать работу компоненты без считывателей, в демонстрационной обработке предусмотрена кнопка «Включить виртуальный режим»:

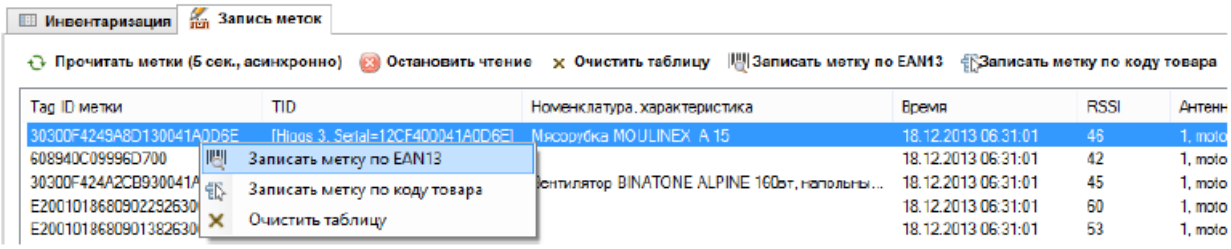


В виртуальном режиме все RFID-считыватели в окне демо-обработки являются фиктивными и компонента на самом деле ни к одному из них не подключается. Все метки, которые будут якобы читаться компонентой (пока она находится в виртуальном режиме) тоже фиктивные:



В стандартной настройке демонстрационной обработки виртуальная инвентаризация читает 2 конкретные метки всегда + иногда еще 0-2 случайные метки.

В фиктивные метки, прочитанные в виртуальном режиме, можно писать.



При этом писаться будут только те две конкретные метки, а в дополнительные случайные метки писать не имеет смысла, т.к. они генерируются случайно и читаются только один раз. Подробнее читайте в статье «Запись меток».

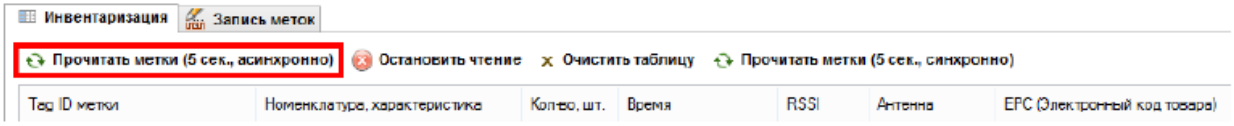
Не нашли что искали?

Задать вопрос в техническую поддержку

Инвентаризация меток с помощью демо-программы

Последние изменения: 2024-03-26

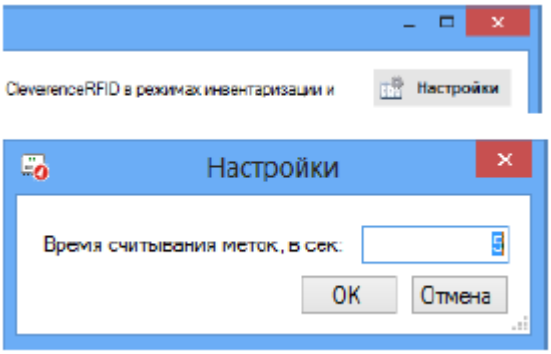
Демонстрационная программа позволяет провести инвентаризацию RFID-меток, находящихся в поле зрения антенн подключенных RFID-считывателей.



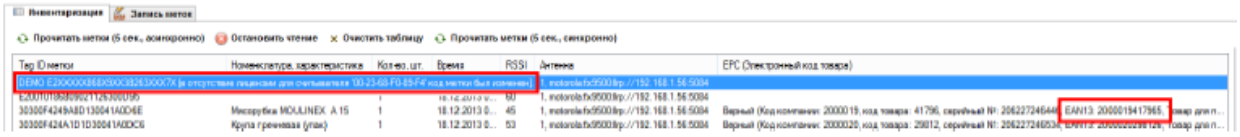
При асинхронном чтении программа не замирает, а метки появляются одна за другой по мере их считывания. Это наиболее удобный режим считывания меток.

При синхронном чтении программа замирает на время чтения, после чего считанные метки появляются сразу все.

Продолжительность времени чтения меток можно задать в настройках программы:



Считанные метки отображаются в таблице, их EPC декодируются:



> В отсутствие лицензии продукта для конкретного считывателя, которым прочитана метка, Tag ID метки может быть изменен на строку «DEMO ...» и текст об отсутствии лицензии.

В приведенной таблице колонка «**Количество**» отображает, сколько меток с идентичной номенклатурой (или одинаковыми Tag ID) было прочитано.

Колонка «**RSSI**» – условный уровень сигнала от метки по шкале от 0 до 100.

В колонке «**Антенна**» показан номер антенны считывателя (от 1 до ....) и URL самого считывателя (если читать сразу с нескольких считывателей).

В поле «EAN13» показан синтетический штрихкод, сгенерированный по данным EPC метки.

Товары ищутся либо по коду товара из EPC, либо по синтетическому штрихкоду EAN13 (подробнее в статье «Запись меток»).

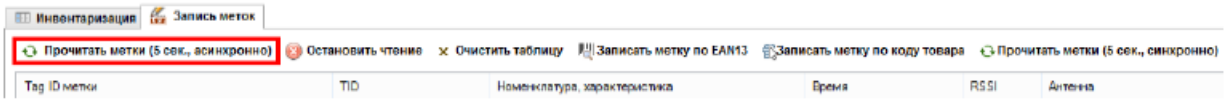
Не нашли что искали?

Задать вопрос в техническую поддержку

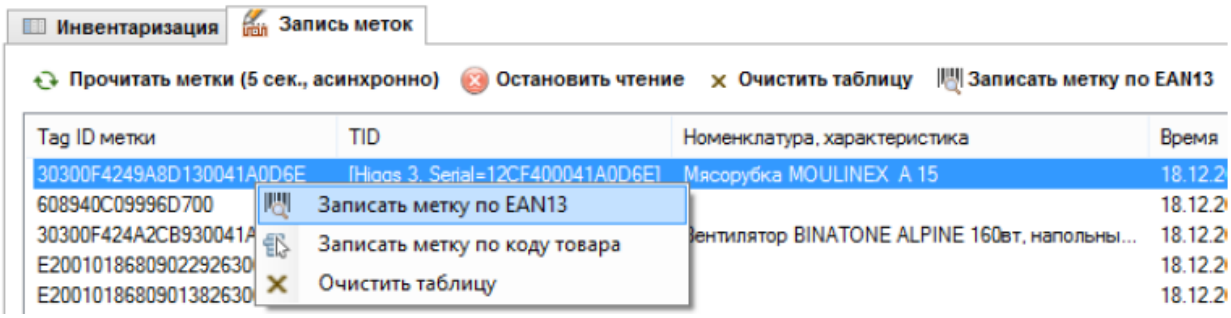
Запись меток с помощью демо-программы

Последние изменения: 2024-03-26

Демонстрационная программа позволяет записать в метки данные о демо-товарах из базы программы. Для этого необходимо переключиться на закладку «Запись меток» и прочитать метки:



Затем можно выбрать любую из прочитанных меток и записать в неё информацию о товаре:



> В отсутствие лицензии на продукт для считывателя, которым прочитана интересующая метка, Tag ID метки может быть заменен на строку «DEMO ...» и текст об отсутствии лицензии. Метки с текстом DEMO недоступны для записи. Однако это не значит, что такую метку совсем нельзя записать – метки заменяются на DEMO в случайном порядке, поэтому можно попытаться еще раз нажать одну из кнопок «Прочитать метки...» до тех пор, пока интересующая метка не будет нормально прочитана.

Запись происходит упрощенно по следующему алгоритму:

1. Обработка генерирует EPC либо на основе штрихкода EAN13, либо по числовому коду товара;
2. Сгенерированный EPC записывается в банк 01 (EPC-bank) выбранной RFID-метки.

Сразу после записи обработка выполняет повторное чтение меток, чтобы можно было увидеть результат записи.

Не нашли что искали?

Задать вопрос в техническую поддержку

Маркировка объектов UHF RFID метками

Последние изменения: 2024-03-26

Под маркировкой объектов понимаются две вещи:

1. Оклейка товаров/ палет/ контейнеров/ документов и т.п. RFID-метками (на которых при этом, возможно, еще и что-то напечатано).
2. Принятые правила, стандарты, решения о том, что писать в метку, что печатать на метке, как это всё оформлять, чтобы маркированный объект было наиболее просто идентифицировать (как визуально, так и с помощью RFID-считывателя).

Как и в случае со штрихкодами, маркировки товаров/ палет/ контейнеров/ документов должны как-то отличаться друг от друга, чтобы помочь оборудованию, программам и сотрудникам избежать ошибок идентификации.

Для UHF RFID-меток не требуется придумывать форматы/ префиксы/ постфиксы штрихкодов. Для всего необходимого уже предусмотрены поля в соответствующих электронных кодах объекта.

Электронный код объекта – это стандартная форма цифровой маркировки объектов при помощи байтов в памяти чипов UHF RFID-меток. Существует два вида электронного кода для UHF RFID: EPC и UII, каждый для своих задач.

Электронный код следует рассматривать как некую карточку, с заранее предусмотренными полями для заполнения. В этой «карточке» могут быть поля для указания кода товара, штрихкода и т.п.

Что можно записать в UHF RFID-метку

В чип UHF RFID-метки можно записать:

1. Электронный код объекта (EPC или UII), по которому будет идентифицироваться то, на что нанесена RFID-метка. Он пишется в EPC/UII-банк памяти, используемый для быстрой инвентаризации и поиска.
2. Дополнительные данные. Они пишутся в банк памяти USER, используемый для получения дополнительных данных у отдельных объектов (работает крайне медленно, а у некоторых чипов может иметь размер ноль байт).
3. Пароли на доступ и «убийство» метки.

Для обеспечения считывания большого числа меток на высокой скорости память UHF RFID-меток искусственно ограничена. Поэтому в первом приближении можно считать, что в UHF RFID-метку помещается информации не больше, чем в обычный одномерный (линейный) штрихкод. И если информацию нельзя уместить в небольшой штрихкод, то в большинстве случаев её нельзя будет записать на UHF RFID-метку.

Второе ограничение связано с тем, что UHF RFID требует обеспечения уникальности маркировки и соответствия международным стандартам, иначе на складе начнутся как свои читаться все подряд чужие и «левые» метки (подробнее читайте в статье «Маркировка объектов при помощи RFID»).

Реально для записи доступно совсем немного памяти RFID-метки + эта память должна быть записана по определенным правилам. Всё это означает, что для задач учета на «воротах», инвентаризации и проверки поступления в метку можно записывать либо только EPC, либо только UII.

Следует иметь в виду, что память EPC/UII для записи UII и память USER для записи дополнительных данных у самых бюджетных RFID-меток обычно крайне ограничена. Фактически, в память USER размером стандартные 32 бита не поместится ни одно стандартное дополнительное поле. Это следует учитывать при разработке системы учета и выборе меток.

Электронный код объекта

EPC содержит информацию о товаре, компании-производителе (или компании-владельце) и серийном номере конкретной единицы каждого товара, объекта или упаковки.

EPC – наиболее предпочтительный способ маркировки объектов для задач логистики и розницы.

Если по каким-либо причинам EPC не подходит для маркировки, альтернативой ему может послужить UII. UII позволяет записать в RFID-метку аналог штрихкода EAN128 или MH10.8.2.

Дополнительные данные

Дополнительные данные, которые пишутся в банк USER RFID-метки, представляют собой аналоги дополнительных штрихкодов EAN128 или MH10.8.2 на упаковке товара, т.е. могут содержать те же самые данные в тех же форматах, что и эти штрихкоды. Низкая скорость считывания банка USER не позволяет массово читать эти дополнительные данные при инвентаризации или на проверке прихода. Фактически подразумевается, что дополнительные данные будут читаться только выборочно, по необходимости, и по очереди у каждой метки в отдельности (например, чтобы отобразить на экране какую-то детальную информацию о товаре или контейнере).

Процедура маркировки

Маркировка при помощи стационарного считывателя

Для маркировки конкретного объекта в общем виде следует выполнить следующие шаги:

1. Нанести «чистую» метку на объект.
2. Поместить объект с нанесенной на него меткой в поле действия антенны считывателя.
3. Сгенерировать соответствующий маркируемому объекту EPC или UII.
4. Прочитать метки вокруг антенны (при этом читать не только Tag ID, но и TID, чтобы иметь возможность однозначно идентифицировать конкретную метку по чипу).
   Для этого используется метод API компоненты
   ПрочестьМетки (<readTime>, <password>, <readTid>, <readUser>, <readReserved>)

Имя параметра	Описание
readTime	Количество времени в миллисекундах, в течение которого считывателю следует искать метки.
password	Пароль доступа, может оказаться необходим для чтения дополнительных банков (RESERVED, TID или USER), по умолчанию ноль.
readTid	Читать ли TID банк меток.
readUser	Читать ли USER банк меток.
readReserved	Читать ли RESERVED банк меток.

5. Убедиться, что читается только одна метка (т.е. что в поле действия нет других меток);
6. Записать сгенерированный EPC или UII в память метки, используя для этого TID:
   для этого используется метод API компоненты

   ЗаписатьEPCUIIпоTID (<tagId>, <tid>, <epcuii>, <accessPassword>)

Имя параметра	Описание
tagId	Tag ID нужной метки для записи.
tid	Содержимое банка TID нужной метки для записи.
epcuii	Записываемый EPC/UII - электронный код товара или уникальный идентификатор объекта.
accessPassword	Число в 32 бита, задающее пароль на доступ к метке. Если пароля нет, то следует указать 0.

Пример: ЗаписатьEPCUIInoTID (метка.TagId, метка.TID, новыйEPC, 0);

Если запись будет неуспешной, то метод API компоненты бросит исключение.

Не нашли что искали?

Задать вопрос в техническую поддержку

Электронный код продукта (EPC)

Последние изменения: 2024-03-26

Понятие электронного кода продукта

Электронный код продукта (EPC, Electronic Product Code) – это способ нумерации конкретных единиц товаров, упаковки, мест хранения, документов и т.д., который используется при маркировке объектов RFID-метками Class 1 Gen 2 по стандарту EPCglobal GS1.

Электронный код продукта является одним из вариантов электронного кода объекта. Второй вариант – это UII.

В отличие от кодов (штрихкодов) EAN13 или ISBN, которые обозначают только номенклатуру или артикул товара, EPC в идеале идентифицирует конкретные экземпляры, т.к. в нём есть место под уникальный серийный номер каждого экземпляра. При этом, в качестве серийного номера никто не запрещает хранить нули или номер целой партии товара.

RFID-метками могут помечаться не только сами товары, но и составные части товара (вложенные в коробку аксессуары), упаковки более высокого уровня (блоки, короба) а также тара (поддоны, контейнеры). В кодах EPC всё это уже предусмотрено.

Помимо товаров RFID-метками с EPC могут помечаться основные средства, логистические объекты, скидочные карточки и много другое.

Что в настоящий момент можно кодировать при помощи EPC:

1. Товары с серийным номером и их части (SGTIN и CPI)
   (включая разные варианты упаковки товара, блочную упаковку, вкладки в коробку, составные части товара, компоненты, аксессуары и т.п.).
2. Контейнеры, палеты (SSCC).
3. Места расположения (SGLN).
4. Возвращаемую или оборачиваемую тару (сосуды, поддоны – GRAI).
5. Оборудование, другое имущество (основные средства – GIAI).
6. Скидочные сертификаты, карты лояльности (GSRN).
7. Документы (GDTI).
8. Объекты для поставки Министерству Обороны США и т.п. (USDOD и ADI).
9. Что-нибудь другое (GID).

Что кодировать + конкретный способ кодирования в терминах EPC называется «схема».

Помимо конкретных объектов, метки могут наноситься на групповую упаковку, еще более групповую упаковку, либо, наоборот, на составные части объекта. Например, EPC для документа с флагом «упаковка» может быть обозначать папку с документами и даже целый ящик с папками.

Что содержит EPC?

Содержимое зависит от того, что им кодируется (см. список выше). При этом любой без исключения код EPC содержит:

- заголовок (который и определяет, что кодируем и как, т.е. так называемую схему);
- так называемое значение фильтра, которое определяет уровень упаковки, на которую нанесена метка;
- номер компании-производителя, компании-владельца или управляющей организации для маркируемого товара/объекта. Номера компаниям назначаются международной организацией GS1.

Регистрационный номер компании в ЮНИСКАН/GS1 – обязательный элемент многих схем.

Для компаний, у которых нет номера в Юнискане, зарезервирован код «4» (как бы «просто какая-то компания»).

Дополнительно к заголовку и значению фильтра, EPC содержит:

1. Для товаров – номер компании, номер (артикул, SKU) товара по каталогу и серийный номер конкретного экземпляра.
2. Для контейнеров – номер компании, серийный номер контейнера.
3. Для мест – номер компании, номер места + дополнительный код.
4. Для возвращаемой и оборачиваемой тары – номер компании, тип тары и серийный номер экземпляра.
5. Для имущества (основных средств) – номер компании, номер основного средства.
6. Для скидочных сертификатов и карт лояльности – номер компании, номер карты или сертификата.
7. Для документов – номер компании, тип документа и серийный номер экземпляра.
8. Для чего-нибудь другого – номер управляющей организации (также выдаваемый GS1), тип объекта и серийный номер конкретного экземпляра.

Так же, как и большинство стандартных штрихкодов, EPC не содержит информации о количестве, размере, весе или цвете товара, и не предусматривает никакого способа её добавить. Для хранения такой расширенной информации можно:

- самостоятельно кодировать код товара;
- хранить в базе серийные номера единиц товара и получать эту информацию по серийному номеру единицы товара из EPC;
- использовать пользовательский банк памяти RFID-метки. EPC хранится в банке памяти под EPC, а под дополнительную информацию отведен отдельный пользовательский банк памяти USER. К сожалению, банк USER не позволит быстро собирать эту информацию во время инвентаризации, инвентаризация существенно замедлится.

Цифровое кодирование EPC

В RFID-метку EPC записывается при помощи нулей и единиц. Перевод EPC в нули и единицы называется бинарным кодированием EPC, которое уже реализовано в компоненте (самим ничего кодировать и декодировать не нужно).

Из метки EPC считывается точно так же в виде нулей и единиц, и чтобы получить из них код компании или серийный номер товара, необходимо произвести декодирование.

Таким образом, один и тот же EPC может быть записан как минимум двумя способами:

1. как число (нули и единицы, шестнадцатеричная запись и т.п.)
2. как осмысленные декодированные данные. Записан, но не закодирован!

Существует еще третий, самый распространенный способ записи EPC – это строка, представляющая собой последовательную запись в 16-ричном формате всех байт бинарно закодированного EPC, и именно в таком виде EPC отображают программы, которые идут с RFID-оборудованием по умолчанию.

Например, если программа прочитала метку «3024000003320C4063A23312», то это значит, что отдельные байты EPC равны:

	1й байт	2й байт	3й байт	4й байт	5й байт	6й байт	7й байт	8й байт	9й байт	10й байт	11й байт	12й байт
16-ричная запись	30	24	00	00	03	32	0C	40	63	A2	33	12

Если декодировать байты данного EPC, то можно получить следующую информацию:

- Схема кодирования – SGTIN (т.е. закодирован код товара с серийным номером);
- Фильтр – «товар для продажи на кассе»;
- Код компании – 6044 (регистрационный номер компании-производителя товара, на который нанесена метка, в реестре международной организации GS1);
- Код товара – 49 (каталожный код товара, на который нанесена метка, в собственной базе компании-производителя товара. например, код товара в базе «1С:Предприятия»);
- Серийный номер – 1671574290 (серийный номер конкретного изделия).

Серийные номера EPC для товаров

Все варианты «товарных» EPC, без исключения, имеют в себе поле для хранения серийного номера того конкретного объекта (товара или упаковки), который маркирован RFID-меткой. Для «коротких» вариантов EPC (например, длиной в 96 бит) поле для серийного номера представляет собой число и всегда чем-то заполнено (по умолчанию нолём). Для «длинных» вариантов EPC серийный номер представляет собой строку из цифр и латинских букв, по умолчанию там пустая строка.

Уникальные серийные номера позволяют отличить один маркированный объект от другого.

Это необходимо, т.к. «голые» метки UHF Gen2 при инвентаризации неотличимы друг от друга.

Пока все метки находятся «в поле зрения» считывателя и более-менее неподвижны (как, например, на кассе), считыватель может однозначно посчитать количество маркированных объектов, даже если все метки прошиты совершенно идентичными кодами.

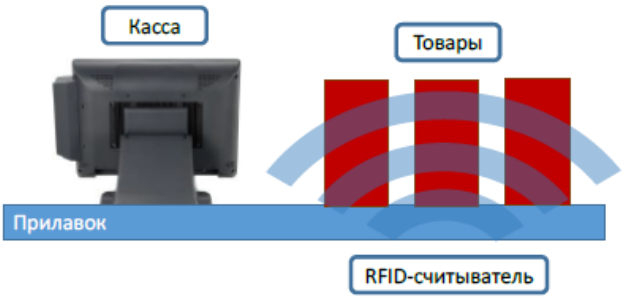
Но во время проведения мобильной инвентаризации либо считыватель, либо объекты, движутся. И метки могут то попадать «в поле зрения» считывателя, то пропадать, то вновь появляться (иногда с разницей в минуты). Что будет, если метки все одинаковые?! В этом случае однозначный подсчет реального количества меток, если у них идентичные коды, просто невозможен, и требуются уникальные серийные номера на каждый объект.

Примеры того, как и почему это плохо, когда у товаров нет уникальных серийных номеров, рассмотрены на примерах ниже.

Таким образом, серийный номера главным образом необходимы для нормальной работы технологии UHF RFID при проведении мобильной инвентаризации товаров.

Без серийного номера

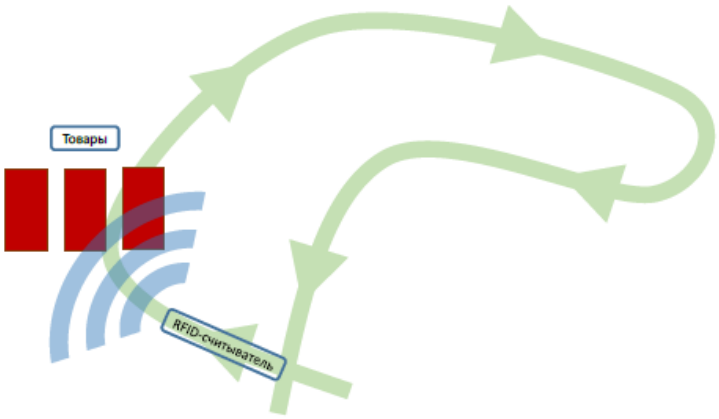
Пример №1: идентичные товары без серийного номера лежат неподвижно.



Без серийного номера

Идентичные товары без серийного номера лежат неподвижно в поле антенн считывателя. Хотя Tag ID меток и идентичны, неотличимы – технология RFID **всё равно позволяет** надежно посчитать точное количество товара.

Пример №2: идентичные товары без серийного номера неподвижны, но считыватель движется.



Без серийного номера

Сначала

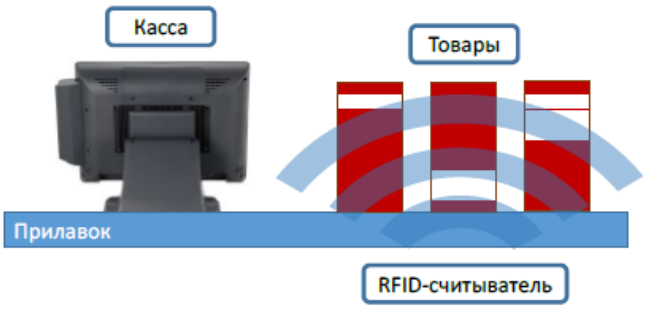
Tag ID меток идентичны, считыватель движется и количество подсвеченных меток «моргает»: то 3, то 2, то 1. В первый раз считыватель насчитает 341 считывание. Сколько товара – неизвестно.

Чуть позже

Tag ID меток идентичны, поэтому, случайно зацепив чуть позже те же самые товары, считыватель «обнаруживает» еще 55 считываний. Итого 395 успешных считываний неизвестного количества товара.

С серийным номером

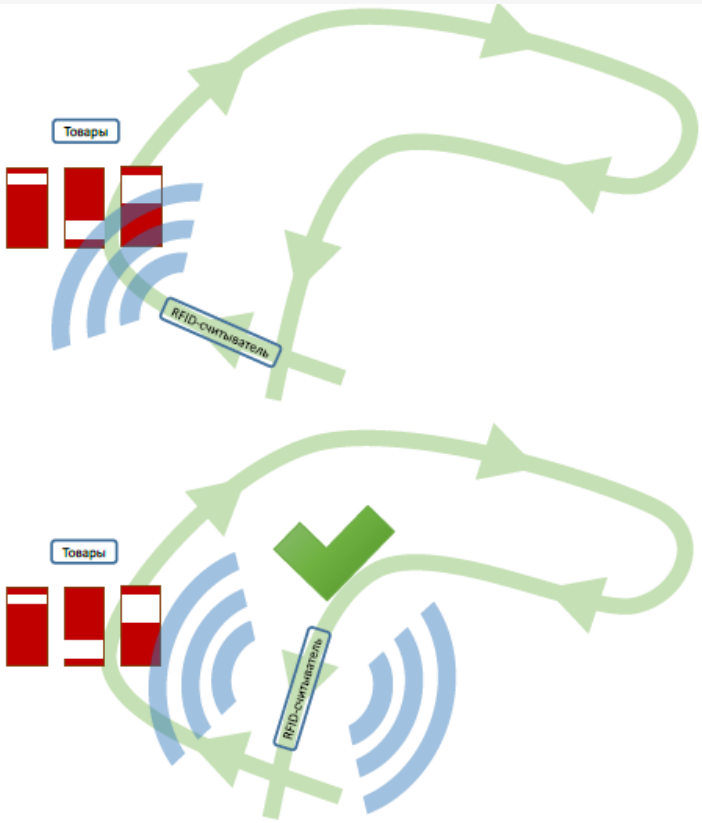
Пример №3: идентичные товары с уникальными серийными номерами лежат неподвижно.



С серийным номером

Товары идентичные, но каждая штучка имеет свой серийный номер. Технология RFID легко позволяет посчитать точное количество такого товара.

Пример №4: идентичные товары с уникальными серийными номерами, считыватель движется.



Генерирование EPC для товаров

Под генерированием EPC понимаются правила, по которым компания будет заполнять поля EPC перед их записью в метку. Для товара нужно заполнить следующие поля: код компании в Юнискан, каталожный код товара, серийный номер, фильтр упаковки (определяет, для чего предназначена метка – для самого товара, для его составной части или для целого палета с товаром). Для компаний, у которых нет кода в Юнискане, зарезервирован код «4» (как бы «просто какая-то компания»).

Данные для заполнения полей берутся либо из учетной системы компании, либо прямо из штрихкодов товаров. Эти правила следует выработать для каждого типа маркируемых товаров, чтобы правильно настроить работу RFID-принтера и/или выделенного маркировочного места со стационарным RFID-считывателем.

EPC по штрихкоду товара

Международные штрихкоды EAN13, ISBN, ISSN, UPC и EAN8, выдаваемые организацией GS1 (в России – Юнискан) для маркировки товаров, журналов и книг на продажу, могут быть переведены в EPC и записаны в метку по схеме SGTIN (товар + серийный номер). К сожалению, это не стандартизовано для внутренних самодельных штрихкодов.

Использование штрихкодов для кодирования RFID-меток позволяет обеспечить легкое и быстрое внедрение RFID-технологии в любой организации, в которой уже используется штрихкодирование товаров.

Преимущества использования стандартных штрихкодов (EAN13, UPC и EAN8, ENA128) для кодирования RFID-меток:

1. Это соответствует международным стандартам.
2. Метки, закодированные таким образом, сможет прочесть любой другой контрагент, использующий стандартные программы и алгоритмы.
3. Нет необходимости менять/ улучшать логистическое ПО, которое уже работает по штрихкодам.
4. Нет необходимости менять кассовое ПО.
5. Нет необходимости менять используемые документы и отчеты (инвентаризационная ведомость и т.п.).

API метод компоненты EPCизШК (EPCfromBarcode) создает экземпляр SGTIN-варианта EPC на основе

штрихкода.

Синтаксис: ЕРСизШК (<barcode >, <filterValue>, <serial>)

Имя параметра	Описание
barcode	Строка со штрихкодом EAN8, EAN13, ISBN, ISSN, UPC или EAN128.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
serial	Серийный номер экземпляра товара, необязательный параметр. Если серийный номер не указан, то его сгенерирует компонента Клеверенс.

Штрихкоды EAN13, UPC и EAN8 для внутреннего использования

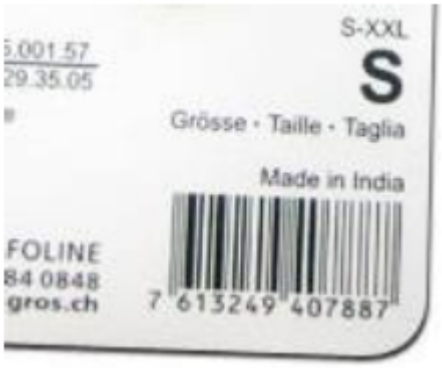
Штрихкоды UPC/EAN, которые начинаются на «04», на «2», а также с «0001…» по «0007…», не являются уникальными и/ или международными. Они предназначены для внутреннего использования в компаниях. Текущая версия стандарта на кодирование UHF меток прямо запрещает использование таких штрихкодов для получения EPC и записи в метку. Конвертация таких штрихкодов в EPC пока не стандартизована, особенно если штрихкод содержит в себе данные о весе, оттенке или размере маркируемого объекта.
Основной вопрос здесь – уникальность получаемых EPC в пределах склада/магазина/офиса, с учетом того, что посетители также проносят на себе какие-то «чужие» метки. На практике он может быть решен также, как и в случае «стандартных» штрихкодов, а именно при помощи уникального серийного номера единицы товара.

Примеры создания EPC товаров на основе штрихкодов EAN для внутреннего использования:

```
epc = ЕРСизШК ("0002490606049", 0, "201216701");
epc = ЕРСизШК ("2000477655015", 0, "201216701");
epc = ЕРСизШК ("2000477655015");
```

Международные штрихкоды EAN13, ISBN, ISSN, UPC и EAN8

Все штрихкоды, которые не подпадают под определение из предыдущего раздела (т.е. почти все штрихкоды из 8, 12 или 13 цифр, которые начинаются не на «0» и не на «2»), представляют собой глобально уникальные международные штрихкоды, которые в теории должны однозначно идентифицировать номенклатуру маркированного объекта (SKU).



Такие штрихкоды, полученные от Юнискан (GS1), могут быть без каких-либо проблем перенесены в UHF RFID-метку при помощи нашего продукта. Такую метку впоследствии сможет прочесть и понять любая стандартная программа в любой точке Земного шара.
Уникальность получаемых EPC в пределах склада/магазина/офиса (с учетом того, что посетители также проносят на себе какие-то «чужие» метки) решается при помощи уникального серийного номера каждой единицы товара (см. «EPC по коду товара»).

Примеры создания EPC товаров на основе международных штрихкодов EAN:

```
epc = ЕРСизШК ("7613249407887", 0, "201216701");

epc = ЕРСизШК ("7613249407887");
```

Штрихкоды EAN128

Штрихкоды EAN128 обычно содержат полный код товара в поле (01) или (02). При генерации EPC при помощи данного продукта штрихкод EAN128 следует использовать целиком, как штрихкод единицы товара.

Уникальность получаемых EPC в пределах склада/магазина/офиса (с учетом того, что посетители также проносят на себе какие-то «чужие» метки) решается при помощи уникального серийного номера каждой единицы товара.
В EAN128 серийный номер содержится в поле (21).
Если в штрихкоде нет полей (01) или (02), то самое разумное - использовать генерацию EPC по коду товара.
Примеры создания EPC товаров на основе штрихкодов EAN128:

```
                    EAN128          о = товар для продажи    серийник
epc = ЕРСизШК ("(01) 80614141123458",        о,              "6789");

epc = ЕРСизШК ("(01) 80614141123458 (21) 6789");
```

Внутренние нестандартные штрихкоды

Самое разумное – забыть про эти штрихкоды и использовать генерацию EPC по коду товара.

EPC по коду товара

API метод компоненты ЕРСизSGTIN (EPCfromSGTIN) создает экземпляр SGTIN-варианта EPC на основе кода компании и кода товара.

Синтаксис: ЕРСизSGTIN (<company>, <item>, <filterValue>, <serial>)

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
item	Код товара согласно каталога компании.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
serial	Серийный номер экземпляра товара, необязательный параметр.

Примеры маркировки товаров при помощи EPC

Пример №1: каждой рубашке назначен свой серийный номер:

Считывателю ответили 3 RFID-метки с EPC равными:
3024000003320C4063A23312 : 1
3024000003320C4063A23313 : 1
3024000003320C4063A23314 : 1

Итого 3 шт.

Согласно информации, закодированной в этих EPC, перед нами три товара *для продажи на кассе*, производства компании под номером «6044», каталожный номер «49», с серийными номерами «1671574290», «1671574291» и «1671574292».

Пример №2: у всех рубашек одинаковые серийные номера (плохо!):

Считывателю ответили 3 RFID-метки с идентичными EPC
3024000003320C4063A23312 : 3

Итого 3 шт.

Согласно информации, закодированной в этих EPC, перед нами три идентичных товара *для продажи на кассе*, производства компании под номером «6044», каталожный номер «49», все три с серийным номером «1671574290».

На кассе это отлично сработает, а с инвентаризацией могут быть проблемы.

Пример №3: на таре, используемой для удобства переноски, есть своя RFID-метка:

Считывателю ответили 3 RFID-метки:

3024000003320C4063A23312 : 1
3084000003320C4063A23312 : 2

Итого товара 1 шт.

Согласно информации, закодированной в этих EPC, перед нами 1 (один) экземпляр *товара для продажи на кассе* + 2 шт. *тары для удобства переноски*, всё производства компании под номером «6044», каталожный номер товара «49», все упаковки с одинаковым серийным номером «1671574290».

Пример №4: ювелирный набор из кулона и двух сережек, на каждом элементе своя метка + отдельная метка самого товара, наклеена на коробочке от набора; проба и другие характеристики ищутся в базе по серийному номеру:

Считывателю ответили 4 RFID-метки:

303000181CE257587E9CA77C : 1
30F000181CE259D87E9CA451 : 1
30F000181CE259D87E9CA452 : 1
30F000181CE25C587E9CF271 : 1

Итого 1 шт. товара, 3 шт. вложенных элементов

Согласно информации, закодированной в этих EPC, перед нами *один товар для продажи на кассе*, внутри которого *лежат три вложенных объекта*. В данном случае коробочка отдельно не учитывается, а метка на коробке является меткой товара. Всё производства компании под номером «12345», каталожный номер товара – «100701». Вложенные объекты имеют каталожные номера «100711» (2 шт.) и «100721» (1 шт.). У каждого свой серийный номер.

Таким образом из примеров видно, что метки можно наносить:

— как на сам товар, так и на его составные части
при этом есть возможность отличить, что за метки мы читаем, и не учитывать метки составных частей в сумме чека (но учитывать их при контроле комплектности)
— на тару, оптовые и любые другие упаковки этого товара.
при этом есть возможность читать только грузовую упаковку, не обращая внимания на много-много индивидуальных меток товара и его составных частей.

Генерирование EPC для коробок, палет

EPC коробки или палеты по штрихкоду

Тара для удобства переноски

В стандартах UHF RFID предусмотрено такое понятие как «тара для удобства переноски», под которым понимается некая неуникальная тара для конкретного товара. RFID-метка на такую тару генерируется по коду или штрихкоду товара, который в ней лежит, только в качестве фильтра при создании EPC указывается не «Товар для продажи на кассе», а «Тара для удобства переноски и транспортировки».

При этом в EPC нет никакой возможности указать количество штук товара в такой таре (но это можно записать в банк USER).

Так же, как и у товара, у такой тары может быть свой серийный номер, который назначается учетной системой или автоматически генерируется продуктом от «Клеверенс».

Примеры создания EPC переносной тары на основе штрихкодов EAN13:

```
                    EAN13                          тип упаковки                              серийник
epc = EPCизШК ("7613249407887", SGTIN_УпаковкаДляТранспортировки, "201216701");

epc = EPCизШК ("7613249407887", SGTIN_УпаковкаДляПереноски, "201216701");

epc = EPCизШК ("7613249407887", SGTIN_УпаковкаДляПереноски);
```

Номерные коробки, палеты и т.п.

Контейнеры, которым выдаются уникальные номера, например коробки с упакованными заказами или поддоны для палет, маркируются при помощи Serial Shipping Container Code (SSCC). Т.е. сам такой контейнер есть «Serial Shipping Container», а его уникальный номер – это его «Code». Обычно это штрихкод, в котором за SSCC отвечают максимум 18 цифр (из них последняя – чексумма).

Если используется EAN128, что SSCC чаще всего хранится в поле (00), например:

Примеры создания EPC номерного контейнера на основе штрихкода EAN128:

```
epc = EPCизШК ("(00) 106141412345678908");

epc = EPCизШК ("(00) 106141412345678908(01)40761324907887");
```

Если же используется не EAN128, а просто Code128 или любая другая кодировка, и в этот штрихкод пишется код контейнера с какими-то префиксами и т.п., то генерацию EPC следует проводить согласно разделу «EPC коробки, палеты и т.п. по уникальному номеру».

EPC коробки, палеты и т.п. по уникальному номеру

Коробки, палеты, емкости могут рассматриваться, с одной стороны, как логистические контейнеры, а с другой – как тара.

Если у палеты или коробки есть уникальный номер, который хранится в учетной системе, возможно участвует в каких-то внутрискладских документах, и определяет содержимое контейнера, то такая коробка или палета считается маркированной при помощи Serial Shipping Container Code (SSCC), т.е. сама коробка или палета есть «Serial Shipping Container», а её уникальный номер – это её «Code». Просто «тарой для удобства переноски» такая коробка или палета не является.

Если у поддона или коробки есть уникальный номер, но он идентифицирует просто оборачиваемую тару, то

такая коробка или палета маркируется при помощи Global Returnable Asset Identifier (GRAI), т.е. сама тара есть «Returnable Asset», а её уникальный номер – «Global Identifier».

Коробка или палета как контейнер

При генерации EPC для такой коробки или палеты следует заполнить два поля: номер компании в Юнискан и уникальный номер конкретного контейнера. Для компаний, у которых нет кода в Юнискане, зарезервирован код «4» (как бы «просто какая-то компания»). Номера контейнеров могут быть только цифровыми, никаких букв и прочих символов. Данные берутся из учетной системы компании, «Wonderfid™ Link» не умеет сам генерировать серийные номера для контейнеров, их следует назначать самостоятельно.

Примеры создания EPC коробок и палет на основе их уникальных номеров:

```
                    Номер компании в Юнискан   Номер коробки или паллеты
epc = EPCизSSCC (4062146,                      7012465122);

epc = EPCизSSCC (4,                            7012465122);
```

Коробка, поддон, пробирка и т.п. как оборачиваемая тара

При генерации EPC для такой коробки или палеты следует заполнить три поля: номер компании в Юнискан, тип оборачиваемой тары (просто цифровой номер, назначаемый самой компанией) и уникальный серийный номер конкретного экземпляра тары. Для компаний, у которых нет кода в Юнискане, зарезервирован код «4» (как бы «просто какая-то компания»). Серийные номера тары могут быть только цифровыми, никаких букв и прочих символов.

Примеры создания EPC оборачиваемой тары:

```
                    Номер компании в Юнискан   Номер типа тары   Номер коробки, поддона или пробирки
epc = EPCизGRAI (4062146,                      1,                7012465122);

epc = EPCизGRAI (4,                            2,                7012888171);
```

Генерирование EPC для документов

Под генерированием EPC понимается правила, по которым компания будет заполнять поля EPC перед их записью в метку. Для документа поля следующие: код компании в Юнискан, тип документа и серийный номер конкретного экземпляра документа. Для компаний, у которых нет кода в Юнискане, зарезервирован код «4» (как бы «просто какая-то компания»).

Данные для заполнения полей берутся из учетной системы компании. Серийные номера документов могут быть только цифровыми, никаких букв и прочих символов. «Wonderfid™ Link» не умеет сам генерировать серийные номера для документов, их следует назначать самим.

Примеры создания EPC документов:

```
                    Номер компании в Юнискан   Номер типа документа   Серийный номер экземпляра
epc = EPCизGDTI (4062146,                      04021,                 44200122213);
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Уникальный идентификатор объекта (UII)

Последние изменения: 2024-03-26

Понятие уникального идентификатора объекта (UII)

Уникальный идентификатор объекта (UII, Unique Item Identifier) – это способ идентификации любых объектов, в том числе логистических, который используется при маркировке объектов любыми RFID-метками по стандартам ISO/IEC.

Уникальный идентификатор объекта является одним из вариантов электронного кода объекта. Второй вариант – это EPC, о котором рассказывается в статье «Электронный код продукта (EPC)».

Что содержит UII?

UII может содержать множество разных полей. Количество полей, записываемых в метку, ограничен только памятью метки.

Для задач логистики и розницы подходит набор полей, описанный в стандарте ANSI MH 10.8.2. Например, наиболее применимые:

Data Identifier	Описание	Формат строки
J	Автомобильный номер	an1+an...35
P	Просто какой-то код объекта, назначаемый покупателем (код для покупятеля)	
1P	Просто какой-то код объекта, назначаемый поставщиком (код от поставщика)	
Q	Количество	
2Q	Вес Нетто	
14D	Дата истечения срока годности	an3+n8
16D	Дата производства	an3+n8
S	Серийный номер	
11K	Номер упаковочного листа	
1K	Номер заказа (назначаемый поставщиком)	
15K	KANBAN	
1H	Код сотрудника (номер сотрудника), назначаемый работодателем	
10K	Номер накладной	
11S	Номер объекта/имущества/основного средства	

Data Identifier – строковой идентификатор поля.

Формат строки расшифровывается следующим образом:

a – заглавная латинская буква

n – любая цифра

an – любые заглавные латинские буквы и/или цифры (вперемешку)

числа, следующие за a/n/an, – требуемое количество указанных символов.

+ просто означает «и далее...»

Примеры разбора формата

Формат	Пояснение	Примеры строк, подходящих под формат
an1+an...35	Одна обязательная буква или цифра, далее от ноля до 35 букв или цифр	«A», «1», «A45», «123132046»
an3+n8	Три обязательных буквы или цифры (вперемешку), далее 8 цифр	«AAA12345678», «00A12345678», «99912345678», «bb0123412345»

Полей UII очень много, они подходят почти для чего угодно, и во многом пересекаются с полями EPC. Но есть две проблемы:

1. **Всё равно нет полей для размеров и цветов одежды.**
2. **Одни и те же поля в UII занимают намного больше памяти, чем в EPC. При прочих равных, в EPC уместится более длинный штрихкод или номер накладной.**

Примеры маркировки при помощи UII

Маркировка автомобилей по автомобильному номеру:

```
uii =UIIизDI("J", "A123MM");          // закодируется в 8 байт
```

Маркировка упаковочного листа (документ за номером 1000141, количество позиций 29):

```
uii =UIIизDI("11K", "1000141");       // номер закодируется в 8 байт
uii.Элемент("Q", "29");               // номер + колво закодируются в 12 байт
```

Маркировка изделия по серийному номеру:

```
uii =UIIизDI("S", "Y7E5N1");          // закодируется в 10 байт
uii =UIIизDI("S", "13235523020537");  // закодируется в 10 байт
```

Маркировка библиотечных объектов RFID метками

Последние изменения: 2024-03-26

Для маркировки библиотечного объекта RFID-меткой следует прошить в метку номер объекта, уникальный в рамках библиотеки.

Маркировка библиотечного фонда

Процедура маркировки книг, журналов и т.п. следующая:

1. Оклеиваем интересующие нас объекты «непрошитыми» RFID-метками;
2. По очереди прошиваем каждую метку соответствующим UII объекта.

Если используется «антикражный бит», то в качестве кода применения («антикражного бита») обязательно выставляем «На складе».

```
// если у библиотеки нет ISIL, то можно передать Неопределено
// если используется «антикражный бит», то AFI = КлеверенсРФИД.AFI.НаСкладе
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, экземпляр.Код, КлеверенсРФИД.AFI.НаСкладе);
```

Если вместо «антикражного бита» используется поиск метки по базе библиотеки и «галочка» в карточке объекта, то в качестве кода применения выставляем «Библиотечный».

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, экземпляр.Код);
// если память метки позволяет, то можно проставить в UII тип использования для объекта
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.ДляВыдачи;
```

UII следует записать в банк EPCUII.

Маркировка читательских билетов (и RFID-карточек)

Читательские билеты можно промаркировать RFID-метками (вклеить в билет), либо полностью заменить билеты RFID-карточками.

Процедура маркировки читательских билетов следующая:

1. Вклеиваем во все читательские билеты «непрошитые» RFID-метками.
2. По очереди прошиваем каждую метку соответствующим UII объекта.

Процедура выдачи RFID-карточек следующая:

1. Вставляем чистую RFID-карточку в специализированный карточный принтер и печатаем на ней фотографию и другую информацию о владельце и библиотеке.
2. Кладем RFID-карточку на антенну считывателя и прошиваем соответствующим UII читателя.

В UII метки для читательского билета желательно указать, что это не книга, а именно читательский билет. Иначе при выдаче и возврате книг об этом придется догадываться по коду билета, читать другие банки памяти и пр.

Код AFI для читательского билета всегда должен быть равен «Библиотечный», чтобы не «звенеть» на воротах

библиотеки и в магазинах.

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, читатель.Код, КлеверенсРФИД.AFI.Библиотечный);
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет;
```

UII следует записать в банк EPCUII.

Предусмотрены следующие типы использования для читательских билетов:

— КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет; (любой)

— КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет_Взрослый;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет_Подростковый;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет_Детский;

```
Если метка.Объект.Тип() = "БиблиотечныйКод" И метка.Объект.ТипИспользования <> Неопределено И
метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет.КодКласса
Тогда
```

Если требуется по логике и позволяет память метки, то можно прошить в банк USER некие дополнительные параметры.

```
// получить с сервера используемый пароль на доступ к RFID-меткам
парольНаДоступ = ПолучитьПарольНаДоступRFID();

бо = КлеверенсРФИД.СоздатьБиблиотечныйОбъект();
бо.Наименование = читатель.ФИО;

банк = бо.СформироватьUSERБанк();
считыватель.ЗаписатьUSER(метка.TagId, банк, парольНаДоступ);
```

Маркировка библиотечного имущества (столы и стулья)

Библиотечное имущество может потребовать два типа RFID-меток: гибкие (для дерева/пластика) и корпусные (для металла). На гибкие метки можно печатать информацию на специализированном этикеточном принтере. На корпусных RFID-метках можно просто писать маркером или использовать самоклеющуюся этикетку (а на этикетку распечатать при помощи того же специализированного этикеточного принтера).

1. Процедура маркировки собственного имущества следующая:
2. печатаем/пишем/наклеиваем на «непрошитые» RFID-метки наименование, инвентарный номер, штрихкод и т.п.;
3. по очереди прошиваем каждую метку соответствующим UII объекта;
4. оклеиваем имущество прошитыми RFID-метками.

В UII метки для имущества желательно указать, что это не книга, а именно библиотечное имущество. Если используется «антикражный бит», то в качестве кода применения («антикражного бита») обязательно выставляем «На складе».

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, имущество.Код, КлеверенсРФИД.AFI.НаСкладе);
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество;
```

UII следует записать в банк EPCUII

Предусмотрены следующие типы использования для имущества:

— КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество; (любое)

— КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество_Компьютер;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество_Видеопроектор;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество_Кинопроектор;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество_Доска;

— КлеверенсРФИД.Библиотеки.ТипыИспользования.НеДляВыдачи;

```
Если метка.Объект.Тип() = "БиблиотечныйКод" И метка.Объект.ТипИспользования <> Неопределено И
(метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество.КодКласса или
метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.НеДляВыдачи.КодКласса) Тогда
```

Если требуется и позволяет память метки, то можно прошить в банк USER некоторые дополнительные параметры.

Общий алгоритм маркировки

Поскольку метки прошиваются конкретным библиотечным кодом, все их следует прошивать по очереди. Наиболее удобный способ – сначала оптом обклеить интересующие объекты «непрошитыми» метками, а затем по одному прошить уникальным кодом.

Алгоритм следующий:

1. По одному кладем объекты на антенну RFID-считывателя.
2. Выбираем из базы, что это такое.
3. Формируем UII на основе некоего уникального используемого в библиотеке кода.
4. Программа должна убедиться, что метка в поле чтения присутствует только одна. Если меток больше – выдать предупреждение.
   Иногда невозможно организовать работу так, чтобы читалась только одна метка. В этом случае программа может опираться на значение RSSI (уровень сигнала от метки) и проверять, что метка с большим RSSI в поле зрения только одна.
5. Прописать сформированный UII в метку. Затем сразу же прочитать метку и проверить, что всё записалось корректно.

```
// получить используемый пароль на доступ к RFID-меткам, если такой используется
// в обычной ситуации парольНаДоступ = 0.
парольНаДоступ = ПолучитьПарольНаДоступRFID();
Пока Истина Цикл
    // Заставить пользователя выбрать из базы конкретный объект фонда, читательский билет и т.п.
    // если выбранному объекту уже сопоставлена метка – переспросить пользователя
    // (например, метка могла выйти из строя и действительно требуется перемаркировка)
    маркируемыйОбъект = ВыбратьЭкземпляр();
    Если маркируемыйОбъект = Неопределено Тогда
        Возврат;
    КонецЕсли;

    режим = РежимДиалогаВопрос.ОКОтмена;
    выбраннаяМетка = Неопределено;
    Пока выбраннаяМетка = Неопределено Цикл
        ответ = Неопределено;
        метки = Неопределено;
        // Поискать вокруг антенны RFID-метки в течение 1й секунды (1000 миллисекунд), при этом читаем не только
        // банк EPCUII, но и банк TID для последующей однозначной идентификации конкретной метки
        Попытка
            метки = считыватель.ИнвентаризоватьМетки(1000, Истина);
        Исключение
            Вопрос("Ошибка поиска меток! " + КлеверенсРФИД.ОписаниеОшибки(), РежимДиалогаВопрос.ОК);
            Продолжить;
        КонецПопытки;

        Если метки.Количество = 0 Тогда
            ответ = Вопрос("Положите маркируемый объект на антенну!", режим);
        Иначе
            Если метки.Количество > 0 Тогда
                ответ = Вопрос("Уберите от антенны посторонные предметы!", режим);
            Иначе
                // Выбрать единственную метку
                выбраннаяМетка = метки.Элемент(0);
            КонецЕсли;
        КонецЕсли;

        Если ответ = КодВозвратаДиалога.Отмена Тогда
            Прервать;
        КонецЕсли;
    КонецЦикла;

    Попытка
        // Создать UII в соответствии с тем, какой объект выбрали, и с правильным AFI:
        uii = СоздатьПравильныйUII(маркируемыйОбъект);
        // Записать UII
        считыватель.ЗаписатьEPCUIIдляTID(выбраннаяМетка.TagId, выбраннаяМетка.TID, uii, парольНаДоступ);

        Сообщить("В метку с Tag ID [" + выбраннаяМетка.TagId + "] успешно записан новый UII [" +
                                    uii.Строка() + "] (" + uii.БинарноеПредставление + ").");
    Исключение
        Предупреждение("Ошибка записи в метку! " + КлеверенсРФИД.ОписаниеОшибки());
    КонецПопытки;
КонецЦикла;
```

Антикражный механизм для библиотек

Стандарт ISO 28560 предлагает на выбор три варианта реализации антикражной системы для библиотеки:

1. **Использовать коды применения (AFI).**
   У всего, что можно выносить, используется код применения «Библиотечный». У всего, что нельзя выносить, – код применения «НаСкладе».
   При выдаче/возврате коды применения в метках перепрошиваются.
   Это наиболее предпочтительный метод, т.к. позволяет RFID-считывателю на антикражных воротах работать автономно без подключения к библиотечной системе, не требует дополнительного оборудования на выдаче/возврате.

2. **Использовать поиск по базе данных.**
   В этом случае RFID-считывателю на антикражных воротах требуется постоянное подключение к библиотечной базе, чтобы искать в ней по UII и смотреть, что выносят.

3. **Использовать гибридные UHF/EAS или HF/EAS метки.**

Т.е. использовать метки, в которых помимо RFID есть антикражная полоска. Метки будут дороже обычных, но это позволяет задействовать обычные магазинные антикражные ворота. Однако, EAS работает на других частотах и, соответственно, для выдачи/возврата потребуется либо гибридное RFID/EAS оборудование (дорогое), либо два набора оборудования и лишние действия при выдаче/возврате.

Решение, какой метод подходит лучше, принимает сама библиотека.

Выдача и возврат книг

При выдаче и возврате объектов фонда программа должна следовать следующему алгоритму:

1. Просканировать пространство вокруг антенны RFID-считывателя на наличие меток.
2. Посмотреть, нет ли среди прочитанных UII читательских билетов. Если их больше одного – попросить убрать лишние и снова просканировать пространство. Если нет ни одного – попросить положить (или заставить библиотекаря выбрать читателя вручную).
3. На основе остальных UII сформировать список выдаваемого/ возвращаемого.
4. Если используется «антикражный бит», то перепрошить его в метках объектов фонда (но не трогать его в читательских билетах, имуществе и прочем вокруг!).

```
// получить с сервера используемый пароль на доступ к RFID-меткам
парольНаДоступ = ПолучитьПарольНаДоступRFID();
типДляВыдачи = КлеверенсРФИД.Библиотеки.ТипыИспользования.ДляВыдачи;
// цикл по всем считанным меткам
Для индекс = 0 по метки.Количество - 1 Цикл
    метка = метки.Элемент(индекс);
    Если метка.Объект.Тип() <> "БиблиотечныйКод" или
      (метка.Объект.ТипИспользования <> Неопределено И метка.Объект.ТипИспользования <> типДляВыдачи)
      Тогда
        Продолжить;
    КонецЕсли;

    Попытка
        // Проставить код применения «Библиотечный» (при выдаче) или «НаСкладе» (при возврате):
        uii.AFI = КлеверенсРФИД.AFI.Библиотечный;
        // Записать UII
        считыватель.ЗаписатьEPCUII(метка.TagId, uii, парольНаДоступ);
    Исключение
        Предупреждение("Ошибка записи в метку! " + КлеверенсРФИД.ОписаниеОшибки());
    КонецПопытки;
КонецЦикла;
```

Настройка виртуального режима работы компоненты в «1С:Предприятие»

Последние изменения: 2024-03-26

Для тестирования работы компоненты без RFID-считывателя на руках, в ней предусмотрен так называемый «виртуальный режим», в котором компонента подключается к виртуальным считывателям и читает виртуальные метки. «Виртуальный» в данном случае означает «отсутствующий на самом деле».

Для активации виртуального режима используется следующий код:

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.Включен = Истина;
```

Настройки виртуального режима позволяют задать параметры работы несуществующих считывателей так, чтобы они удовлетворяли условиям проводимых тестов.

Пример №1 | виртуальное чтение всегда ровно 6-ти случайных меток.

При такой настройке генерируются шесть случайных меток, которые будут «прочитаны».

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 6;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 0;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

Пример №2 | виртуальное чтение от 6-ти до 10-ти случайных меток.

В такой настройке компонента будет от инвентаризации к инвентаризации генерировать от шести до десяти случайных меток.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 6;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 10;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

Пример №3 | виртуальное чтение двух заранее заданных меток.

В такой настройке компонента всегда будет «читать» только две указанные метки.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 2;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 2;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000001");
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000002");
```

Пример №4 | виртуальное чтение двух заранее заданных и одной-двух случайных меток.

В такой настройке компонента от инвентаризации к инвентаризации будет «читать» либо две указанные метки + одна случайная, либо две указанные + две случайных.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 3;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 4;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000001");
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000002");
```

Пример №5 | виртуальное чтение трех заранее заданных и нескольких случайных меток.

В такой настройке компонента от инвентаризации к инвентаризации будет генерировать от ноля до семи случайных меток и «читать» их наряду с тремя заранее заданными.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 3;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 10;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();

// создаем метку по Tag ID.
tagid1 = "300800000000000000000000";
метка1 = КлеверенсРФИД.НоваяМетка(tagid1);
// Аттрибут «Счетчик» означает число меток с идентичным EPC.  Если Счетчик = 2, то при инвентаризации были
// обнаружены две метки с идентичным EPC.  В реальной инвентаризации вместо того, чтобы возвратить две
// одинаковые метки, компонента объединяет их в одну, и проставляет счетчик = 2.
метка1.Счетчик = 2;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка1);

// создаем EPC единицы товара с серийным номером «4412»,  кодом товара «123» от фирмы с кодом «7770».
// первый ноль означает, что это EPC товара для продажи на кассе.
epc = КлеверенсРФИД.EPCизSGTIN(0, 7770, 123, "4412");
// создаем метку по EPC.
метка2 = КлеверенсРФИД.НоваяМетка(epc);
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка2);

// создаем EPC единицы товара с серийным номером «332», сам товар задаем по EAN13.
// первый ноль означает, что это EPC товара для продажи на кассе.
epc = КлеверенсРФИД.EPCизEAN13(0, "4004764390793", "332");
// создаем метку по EPC.
метка3 = КлеверенсРФИД.НоваяМетка(epc);
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка3);
```

Пример №6 | сначала какое-то время виртуально читается только одна метка, затем только другая

В некоторых ситуациях для тестирования алгоритмов учета может понадобиться управлять сценарием считывания меток. Например, чтобы сразу после запуска инвентаризации читались какие-то одни определенные метки, а спустя пару секунд – другие определенные метки. В приведенной ниже настройке от инвентаризации к инвентаризации компонента будет воспроизводить один и тот же сценарий: сначала «читается» метка "300800000000000000000001", затем она исчезает и начинает «читаться» метка "300800000000000000000002".

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 2;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 2;

ТестовыеМетки = КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки;

// добавляем метку по Tag ID.  Метка начинает читаться спутся примерно 1 сек. и видна примерно 5 сек.
ТестовыеМетки.ДобавитьПоВремени("300800000000000000000001", 1, 5);
// добавляем метку по Tag ID.  Метка начинает читаться на 8й сек. и видна примерно 2 сек.
ТестовыеМетки.ДобавитьПоВремени("300800000000000000000002", 8, 2);
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Обработка внешних событий в «1С:Предприятии»

Последние изменения: 2024-03-26

По мере работы компоненты в предопределенную процедуру «ОбработкаВнешнегоСобытия» основного модуля «1С:Предприятия», а также в процедуру «ВнешнееСобытие» формы приходят события.

Источник = Строка "CleverenceRFID"

Событие = Наименование события

Данные = Данные, связанные с событием

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" Тогда
    // обработать событие от компоненты
  КонецЕсли;
КонецПроцедуры
```

Всего в настоящий момент компонента может генерировать 4 события:

1. НайденСчитыватель.
2. Чтение.
3. ЧтениеОкончено.
4. Запись.

Более подробно о каждом из событий написано далее.

Поиск и подключение RFID считывателей

Возможности компоненты позволяют производить поиск RFID-считывателей в локальной подсети (т.е. в диапазонах IP-адресов «192.168.0.1 – 192.168.248.255», «172.16.0.1 - 172.16.240.255» и «10.0.0.1 - 10.255.255.255»). К сожалению, текущая версия поиска работает только внутри небольших сетей из 5-20 компьютеров и в подсетях 255.255.255.* (т.е. если у вас задана слишком широкая подсеть, то поиск скорее всего не сработает).

По физическому подключению и настройке RFID-считывателей «Установка и настройка RFID считывателей»».

Синхронный поиск считывателей

При синхронном поиске окна «1С:Предприятия» замирают на время выполнение процедуры «НайтиСчитыватели» компоненты (примерно 20-30 сек).

Пример кода синхронного поиска считывателей:

любой модуль:

```
считыватели = КлеверенсРФИД.НайтиСчитыватели();
Для индекс = 0 по считыватели.Количество - 1 Цикл
    считыватель = считыватели.Элемент(индекс);
    // Сообщить url найденного RFID-считывателя:
    Сообщить("Найден считыватель: " + считыватель.Url);
КонецЦикла;
```

Асинхронный поиск считывателей

При асинхронном поиске окна «1С:Предприятия» не замирают, т.к. поиск выполняется в фоне. По мере нахождения новых считывателей, компонента посылает внешнее событие «НайденСчитыватель», которое можно обработать в главном модуле.

любой модуль:

```
КлеверенсРФИД.НачатьПоискСчитывателей();
```

Событие «НайденСчитыватель»

При асинхронном поиске новых считывателей в локальной подсети, компонента посылает внешнее событие «НайденСчитыватель».

Источник = "CleverenceRFID"

Событие = "НайденСчитыватель"

Данные = Url найденного считывателя, например «motorola:xr480:llrp://10.10.0.17».

Подключиться к найденному считывателю по полученному url можно позднее, используя метод компоненты «ПодключитьСчитыватель».

По мере работы компоненты в предопределенную процедуру «ОбработкаВнешнегоСобытия» основного модуля «1С:Предприятия», а также в процедуру «ВнешнееСобытие» формы приходят события.

Источник = Строка "CleverenceRFID"

Событие = Наименование события

Данные = Данные, связанные с событием

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" Тогда
    // обработать событие от компоненты
  КонецЕсли;
КонецПроцедуры
```

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "НайденСчитыватель" Тогда
    // Сообщить url найденного RFID-считывателя:
    Сообщить("Найден считыватель: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

Или, если подписать форму на событие «ВнешнееСобытие»:

Модуль формы:

```
Процедура ВнешнееСобытие(Источник, Событие, Данные)
  Если Источник = "CleverenceRFID" И Событие = "НайденСчитыватель" Тогда
    // Сообщить url найденного RFID-считывателя:
    Сообщить("Найден считыватель: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Объект «RFID метка» в «1С: Предприятии»

Последние изменения: 2024-03-26

В рамках продукта RFID-метка представляет собой специальный объект с набором реквизитов и функций. Наиболее подробно все объекты рассмотрены в статье «Справочник разработчика».

Реквизиты объекта компоненты RFID-метка («Cleverence.RFID.RfidTag»)
Имя реквизита
Имя реквизита англ.
Описание
TagId
TagId
Возвращает Tag ID метки 16-ричном представлении (строка в 24 символа).
Считыватель
Reader
Возвращает считыватель, при помощи которого была считана данная метка.
Объект
Identity
Возвращает декодированное значение электронного кода объекта (EPC или UII) прочитанной метки.
НомерАнтенны
AntennaId
Возвращает номер (код) антенны, которая прочла метку с таким Tag ID.
Время
FirstTimeSeen
Возвращает дату/время, в которое метка с таким Tag ID была увидена впервые (по часам компьютера, на котором работает компонента).
Счетчик
SeenCount
Возвращает сколько раз была замечена метка с таким Tag ID. Фактически, для неподвижно лежащих меток это число отражает количество меток с разным номером чипа (TID), но одинаковым Tag ID (одинаковым EPC/UII). Для движущихся меток сюда добавляется количество входов/выходов таких меток за пределы области чтения.
RSSI
PeakRSSI
Возвращает пиковое значение принятого уровня сигнала от метки в произвольных единицах от 0 до 255 (число).
RESERVED
RESERVED
Возвращает декодированное содержимое банка RESERVED (если оно было прочитано командой) либо «Неопределено», если банк недоступен или не читался.

TID
TID
Возвращает декодированное содержимое банка TID (если оно было прочитано командой) либо «Неопределено», если банк недоступен или не читался.
USER
USER
Возвращает декодированное содержимое банка USER (если оно было прочитано командой) либо «Неопределено», если банк отсутствует, недоступен или не читался.

Не нашли что искали?

Задать вопрос в техническую поддержку

Настройка чтения RFID-меток в «1С:Предприятие»

Последние изменения: 2024-03-26

Операция инвентаризации поддерживается на уровне радио-протокола обмена между метками и считывателем, и возвращает данные о том, какие EPC присутствуют в зоне считывания.

Например, все метки могут иметь один и тот же EPC/UII, и в этом случае по итогам инвентаризации мы будем знать, что это за EPC, и сколько всего RFID-меток с этим EPC/UII удалось считать ридеру.

Если все метки имеют свой уникальный EPC/UII (не путать с уникальным номером чипа, который безусловно есть у каждой метки Class 1 Gen 2), то операция инвентаризации вернет список этих EPC/UII.

Синхронное чтение (инвентаризация) меток

Синхронная инвентаризация означает следующее:

1. «1С:Предприятие» дало считывателю команду «считай окружающие метки в течение N секунд» и замерло в ожидании ответа.
2. Считыватель читает метки, «1С:Предприятие» ждет, все формочки замерли. Считыватель закончил через указанное время и вернул результат «1С:Предприятию».
3. «1С:Предприятие» получило результат, обработало его, формочки «отвисли».

Таким образом, если при синхронной инвентаризации указать считывателю «считай 50 секунд», то окно 1С почти целую минуту не будет доступно для пользователя.

Пример кода для синхронной инвентаризации:

Модуль формы:

```
// ---------- по нажатии кнопки 1 ----------
// Опрашивать окружающие метки в течение 5000 миллисекунд (5 сек)
метки = считыватель.ПрочестьМетки (5000);
Для индекс = 0 по метки.Количество - 1 Цикл
        метка = метки.Элемент(индекс);
        ОбработатьМетку(метка); // Какая-то процедура обработки метки
КонецЦикла;
```

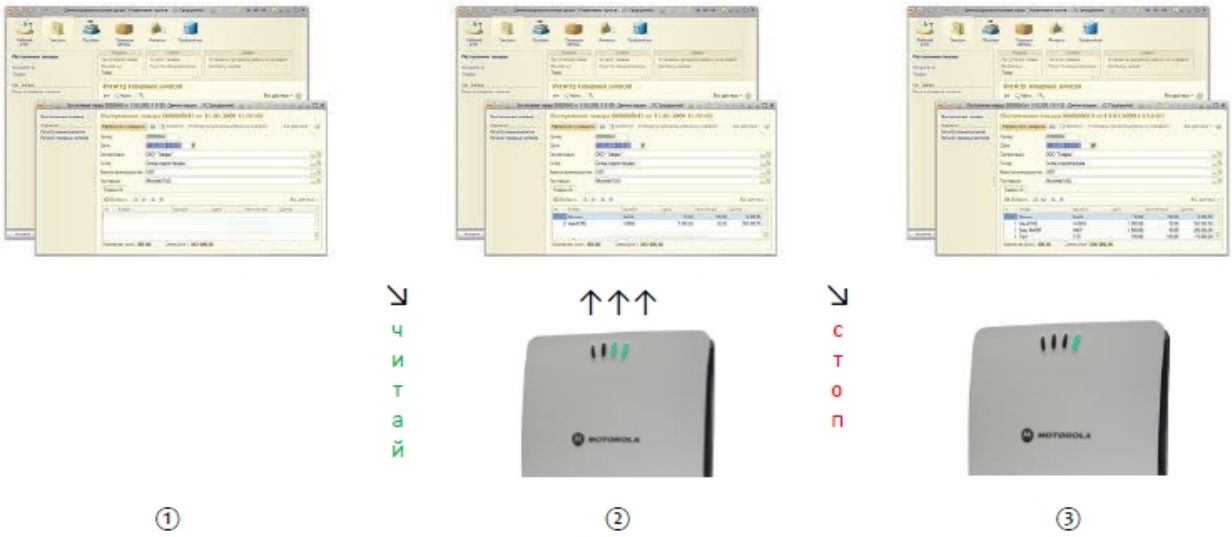
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

> Во время синхронной инвентаризации внешнее событие «Чтение» не приходит, т.к. это «убило» бы приложение 1С.

Асинхронное чтение (инвентаризация) меток

Асинхронная инвентаризация означает следующее:

1. «1С:Предприятие» дало считывателю команду «считай окружающие метки в течение N секунд» и продолжило делать свои дела.
2. По мере инвентаризации новых меток считыватель асинхронно посылает «1С:Предприятию» внешние события, в результате чего считанные метки могут интерактивно появляться в окнах и документах «1С:Предприятия».
3. Считыватель либо закончил через указанное время, либо «1С:Предприятие» дало ему команду закончить инвентаризацию досрочно.



Таким образом, при асинхронной инвентаризации окно 1С всегда остается доступным для взаимодействия с пользователем, а найденные метки могут интерактивно появляться на экране.

Пример кода для асинхронной инвентаризации:

```
// Опрашивать окружающие метки в течение 5000 миллисекунд (5 сек)
КодКомандыСтрока = считыватель.НачатьЧтение(5000);

// Получить все метки, обнаруженные во время инвентаризации (включая и те, по которым приходили события)
метки = считыватель.ОкончитьЧтение();
Для индекс = 0 по метки.Количество - 1 Цикл
    метка = метки.Элемент(индекс);
    ОбработатьМетку(метка);
КонецЦикла;
```

> На одном и том же считывателе нельзя одновременно запускать две и более чтений меток!
> Но при этом разрешается проводить несколько параллельных чтений, если они выполняются на разных считывателях.

Событие «Чтение»

При каждом удачном асинхронном чтении RFID-метки (в частности, при асинхронной инвентаризации) компонента посылает внешнее событие «Чтение».

Источник = "CleverenceRFID"

Событие = "Чтение"

Данные = Строка из номера задания (созвращается методом НачатьЧтение), url считывателя и Tag ID прочитанной метки, через символ ‘@’. Например, «F16828D7-A33D-4320-8D6F-4D8598BCB5EA@motorola:xr480:llrp://10.10.0.17@303000181CE257587E9CA77C».

Более подробную информацию о самой метке можно получить у конкретного считывателя или у самой компоненты через метод «ВыбратьМетку».

В качестве данных в событие приходит только Tag ID метки. Получить более подробные данные можно при помощи метода компоненты «ВыбратьМетку».

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "Чтение" Тогда
    Попытка
        // Получить полные данные считанной метки (или одинаковых меток) сразу со всех считывателей:
        метка = КлеверенсРФИД.ВыбратьМетку(Данные);
        // Либо получить данные у конкретного считывателя (подробнее о считывателях см. ниже)
        // метка = считыватель.ВыбратьМетку(tagid);

        Сообщить(метка.TagId + ", кол-во: " + метка.Счетчик +
                    " шт., время='" + метка.Время.Строка() + "', RSSI=" + метка.RSSI);
          ...
    Исключение
        Сообщить(КлеверенсРФИД.ОписаниеОшибки());
    ОкончаниеПопытки;
  КонецЕсли;
КонецПроцедуры
```

либо, если подписать форму на событие «ВнешнееСобытие»:

ВнешнееСобытие ВнешнееСобытие

Модуль формы:

```
Процедура ВнешнееСобытие(Источник, Событие, Данные)
  Если Источник = "CleverenceRFID" И Событие = "Чтение" Тогда
    Попытка
        // Работа с компонентой
        // Получить полные данные считанной метки (или одинаковых меток) сразу со всех считывателей:
        метка = КлеверенсРФИД.ВыбратьМетку(Данные);
        // Либо получить данные у конкретного считывателя (подробнее о считывателях см. ниже)
        // метка = считыватель.ВыбратьМетку(tagid);

        Сообщить(метка.TagId + ", кол-во: " + метка.Счетчик +
                    " шт., время='" + метка.Время.Строка() + "', RSSI=" + метка.RSSI);
          ...
    Исключение
        Сообщить(КлеверенсРФИД.ОписаниеОшибки());
    ОкончаниеПопытки;
  КонецЕсли;
КонецПроцедуры
```

Событие «ЧтениеОкончено»

При каждом окончании синхронного или асинхронного чтения RFID-меток (как штатном, так и по ошибке) компонента посылает внешнее событие «ЧтениеОкончено».

Источник="CleverenceRFID"

Событие="ЧтениеОкончено"

Данные=Строка причины остановки плюс URL того считывателя, который закончил чтение.

Причины остановки:

«ИстеклоВремя» – закончилось время, указанное при вызове метода чтения меток,

«Оборвано» – метод ОкончитьЧтение() был вызван до того, как истекло время,

«Исключение» – при попытке чтения произошло исключение. Подробности исключения можно посмотреть, вызвав метод ПолучитьОшибку().

Пример данных для события ЧтениеОкончено:

«ИстеклоВремя@motorola:fx9500:llrp://10.10.0.121:5084»

«Оборвано@motorola:fx9500:llrp://10.10.0.121:5084»

В качестве данных в событие приходит специальная строка, в которой через символ «@» указаны причина остановки чтения и URL считывателя, на котором остановлено чтение.

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "ЧтениеОкончено" Тогда
        Сообщить("Чтение окончено: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

либо, если подписать форму на событие «ВнешнееСобытие»:

ВнешнееСобытие ВнешнееСобытие

Модуль формы:

```
Процедура ВнешнееСобытие(Источник, Событие, Данные)
  Если Источник = "CleverenceRFID" И Событие = "ЧтениеОкончено" Тогда
      Сообщить("Чтение окончено: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

Чтение банка EPC/UII

Чтение банка EPC/UII происходит во время инвентаризации меток (которая не требует паролей), а также при чтении любых других банков, поэтому отдельно чтением банка EPC/UII озадачиваться необязательно.

Чтение банка USER

Банк USER хранит любую дополнительную информацию в формате ISO 15961 (конкретные упакованные поля со строковыми значениями) либо просто байтами. В зависимости от используемого в метке чипа, банк USER может быть размером от нуля бит до нескольких килобайт.

Пример №1:

Любой модуль:

```
// Прочесть банки USER всех меток в поле видимости считывателя, в течение 2,5 секунд (2500 миллисекунд)
метки = считыватель.ПрочестьБанкUSER(2500);
Для индекс = 0 по метки.Количество - 1 Цикл
        метка = метки.Элемент(индекс);
        Сообщить("Прочитано: " + метка.tagID + ", USER = " + Строка(метка.БанкUSER));
КонецЦикла;
```

Пример №2:

Любой модуль:

```
// Прочесть банк USER у первой же метки, Tag ID которой равен указанному.
банкTID = считыватель.ПрочестьБанкUSER("30240000033320C4063A23312");
Сообщить("Прочитано: USER = " + метка.БанкUSER.Строка());
```

Чтение банка TID (запись в него невозможна)

Банк TID хранит уникальный номер чипа. Перепрошить этот номер чипа никак нельзя. Если при маркировке объектов вести реестр всех использованных чипов, то банк TID можно использовать для проверки того, что метка не была «заменена злоумышленником».

Пример №1:

Любой модуль:

```
// Прочесть банки TID всех меток в поле видимости считывателя, в течение 1,5 секунд (1500 миллисекунд)
// пароль на доступ = 0 (нет пароля).
метки = считыватель.ПрочестьБанкTID(1500, 0);
Для индекс = 0 по метки.Количество - 1  Цикл
        метка = метки.Элемент(индекс);
        Сообщить("Прочитано: " + метка.tagID +
                                        ", MDID = " + метка.БанкTID.MDID + ", TMN = " + метка.БанкTID.TMN);
КонецЦикла;
```

Пример №2:

Любой модуль:

```
// Прочесть банк TID у первой же метки, Tag ID которой равен указанному.  Пароль на доступ = 0 (нет пароля).
банкTID = считыватель.ПрочестьБанкTID("30240000033320C4063A23312", 0);
Сообщить("Прочитано: MDID = " + банкTID.MDID + ", TMN = " + банкTID.TMN);
```

Чтение банка RESERVED

Банк RESERVED хранит пароли на доступ и блокирование метки. Если метки используются только внутри организации и никуда не передаются, то в целях защиты от несанкционированного перепрошивания меток сторонними лицами всегда имеет смысл установить единый секретный пароль хотя бы на доступ к чтению/записи.

Поскольку на чтение банка RESERVED нужно знать пароль доступа, то большого смысла в операции чтения содержимого банка RESERVED ради пароля доступа нет. Однако, некоторые производители включают в банк RESERVED дополнительную информацию, например альтернативный пароль доступа с которым читается второй «приватный» набор банков (что позволяет организовать «публичную» и «внутреннюю» версии данных одной и той же метки), антикражный флаг и т.п.

Пример:

Любой модуль:

```
// Прочесть банк RESERVED у первой же метки, Tag ID которой равен указанному. Пароль на доступ = 123.
банкRESERVED = считыватель.ПрочестьБанкRESERVED("30240000033320C4063A23312", 123);
Сообщить("Прочитано: пароль доступа = " + банкRESERVED.ПарольДоступа +
        ", пароль на блокирование = " + банкRESERVED.ПарольНаБлокирование);
дополнительныеПароли = банкRESERVED.ДополнительныеБайты;
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Настройка записи RFID-меток в «1С:Предприятие»

Последние изменения: 2024-03-26

Операция записи банка поддерживается на уровне радио-протокола обмена между метками и считывателем и позволяет переписать всю или часть информации в интересующем банке RFID-меток (если эту память не прожгли намертво). В рамках одного запроса можно писать в любое количество банков и любое количество меток одновременно. Считыватель отправляет запрос, а метки, подходящие под условия запроса, каждая по очереди записывается.

Запись сразу в несколько меток

Из 4х банков меток Gen2 для записи доступны три: банк с паролями, банк EPC и пользовательский банк.

Текущая реализация компоненты такова, что записать что-либо в метку можно только зная её Tag ID (чтобы не писать непонятно что в случайные метки). Поэтому прежде чем что-нибудь записать, сначала следует инвентаризировать метки и получить их Tag ID.

Зная Tag ID, можно записать что-нибудь одновременно во все метки с таким Tag ID.

Любой модуль:

```
Попытка
        // Создать EPC:
        epc = ...

        // Записать EPC:
        ПодключенныйСчитыватель.ЗаписатьEPCUII(ИнтересуемаяМетка.TagId, новыйEPC, 0);
        Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "] успешно записан новый EPC [" +
                                    новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
                                                                КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Запись только в одну конкретную метку

Запись только в одну конкретную метку опирается на то, что у каждой метки должен быть свой уникальный номер чипа.

Не зная TID можно просто прочесть банки TID всех меток вокруг и потом записать в нужную:

Любой модуль:

```
Попытка
        новыйEPC = ...

        // Читать метки и банки TID всех меток вокруг в течение 1,5 сек (1500 миллисекунд)
        // пароль на чтение = 0 (нет пароля)
        // возвратится коллекция меток, в каждой из которых будет проставлен реквизит TID
        метки = ПодключенныйСчитыватель.ПрочестьМетки(1500, Истина, Ложь, Ложь);
        // Записать новый EPC по номеру чипа, пароль на запись = 0 (нет пароля):
        ПодключенныйСчитыватель.ЗаписатьEPCпоTID(метки[0].TagId, метки[0].TID, новыйEPC, 0);
        Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "] успешно записан новый EPC [" +
                                        новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
                                                                КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Зная EPC, можно прочитать банк TID одой единственной метки и затем записать только в неё:

Любой модуль:

```
Попытка
        новыйEPC = ...

        // Прочесть номер чипа, пароль на чтение = 0 (нет пароля):
        tid = ПодключенныйСчитыватель.ПрочестьБанкTID(ИзвестныйTagID, 0);
        // Записать новый EPC по номеру чипа, пароль на запись = 0 (нет пароля):
        ПодключенныйСчитыватель.ЗаписатьEPCпоTID(ИзвестныйTagID, tid, новыйEPC, 0);
        Предупреждение("В метку с tag ID [" + ИзвестныйTagID + "] успешно записан новый EPC [" +
                                        новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИзвестныйTagID + "]: " +
                                                                КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Событие «Запись»

При каждой удачной асинхронной записи RFID-метки компонента посылает внешнее событие «Запись».

Источник="CleverenceRFID"

Событие="Запись"

Данные=Старый Tag ID + новый Tag ID через знак «@».

Например,

«303000181CE257587E9C000@303000181CE257587E9CA77C»

Более подробная информация недоступна, метод «ВыбратьМетку» не применим.

В качестве данных в событие приходит только старый и новый Tag ID метки.

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "Запись" Тогда
    // Сообщить Tag ID записанной метки:
    Сообщить("Записана метка: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Для задач логистики и розницы

Последние изменения: 2024-03-26

> Более подробно о том, что такое EPC и зачем он нужен, читайте в разделе «UHF RFID для логистики и розницы».

Цифровое кодирование EPC

В RFID-метку EPC записывеатся при помощи нулей и единиц. Перевод EPC в нули и единицы называется бинарным кодированием EPC, которое уже реализовано в продукте (самим ничего кодировать и декодировать не нужно). Самый распространенный способ записи EPC – это строка, представляющая собой последовательную запись в 16-ричном формате всех байт бинарно закодированного EPC, и именно в таком виде EPC отображают программы, которые идут с RFID-оборудованием по умолчанию.

Строки вида «30240000033320C4063A23312», которые выдаются демопрограммой считывателя как TAG ID считанных RFID-меток, требуют декодирования в EPC. Только тогда можно узнать код товара/документа и т.п., на который нанесена метка. Все стандартные схемы кодирования/декодирования уже реализованы в «Wonderfid™ Link».

Пример декодирования EPC при помощи «Wonderfid™ Link»:

```
epc = КлеверенсРФИД.ДекодироватьEPCUII("30240000033320C4063A23312");
```

Пример кодирования EPC при помощи «Wonderfid™ Link»:

```
строка = КлеверенсРФИД. EPCизSGTIN(0, КодКомпании, КодТовара, 12345).БинарноеПредставление;
```

Генерирование EPC для товаров

Если метки используются для целей контроля за движением товаров/объектов/документов, то самым главным в RFID-метке будет являться банк EPC. В банке EPC/UII будет содержаться собственно EPC или UII, описывающий, на какой конкретно объект будет нанесена RFID-метка.

Под генерированием EPC понимается правила, по которым компания будет заполнять поля EPC перед их записью в метку. Данные для заполнения берутся либо из 1С, либо прямо из штрихкодов товаров. Эти правила следует выработать для каждого типа маркируемых объектов, чтобы правильно настроить работу RFID-принтера и/или выделенного маркировочного места со стационарным RFID-считывателем.

Серийные номера EPC для товаров

Все варианты «товарных» EPC, без исключения, имеют в себе поле для хранения серийного номера того конкретного объекта (товара или упаковки), который маркирован RFID-меткой. Для «коротких» вариантов EPC (например, длиной в 96 бит) поле для серийного номера представляет собой число и всегда чем-то заполнено (по умолчанию нулём). Для «длинных» вариантов EPC серийный номер представляет собой строку из цифр и латинских букв, по умолчанию там пустая строка.

> Более подробно о том, зачем вообще товарам серийные номера, можно прочитать в разделе «UHF RFID для логистики и розницы».

Выдача учетной базой уникальных серийных номеров каждому экземпляру продаваемого товара – задача не из простых. Особенно для сетевых компаний. К счастью, продукт «Клеверенс» уже содержит в себе алгоритмы генерации уникальных серийных номеров, разработанные согласно рекомендациям всех основных

производителей RFID-чипов.

Эти алгоритмы обеспечивают глобальную уникальность генерируемых серийных номеров в диапазоне 3-25 лет (в зависимости от марки чипа, используемого в RFID-метке).

Генерация серийных номеров при помощи «Wonderfid™ Link» выполняется перед записью EPC в метку в том случае, если это EPC товара и в нём не указан серийный номер.

В примерах ниже показано, что серийный номер является необязательным аргументом функций API компоненты. Если серийный номер не передавался или было передано значение «Неопределено», то в созданном EPC он будет пустым и, соответственно, компонента сгенерирует его перед записью в метку.

EPC по штрихкоду товара

«Wonderfid™ Link» предоставляет несколько способов создания EPC на основе штрихкодов товаров. В примерах ниже ШК – любой штрихкод EAN8, EAN13, ISBN, ISSN или UPC.

Пример 1. Товар для продажи на кассе, серийные номера генерирует 1С:

```
// EPC товара на основе штрихкода и уникального серийного номера единицы товара
epc = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерирует «Wonderfid™ Link»:

```
// EPC товара только на основе штрихкода (уникальный серийный номер будет сгенерирован
// компонентой при записи в метку, см. в разделе «Ошибка! Источник ссылки не найден.»)
epc = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы, Неопределено);
// либо (то же самое)
epc = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// либо (то же самое)
epc = КлеверенсРФИД.EPCизEAN13(ШК);
```

EPC по коду товара

API метод компоненты EPCизSGTIN (EPCfromSGTIN) создает экземпляр SGTIN-варианта EPC на основе кода компании и кода товара.

Синтаксис: **EPCизSGTIN (<company>, <item>, <filterValue>, <serial>)**

Имя параметра	Описание
company	Код компании, зарегистрированной в GS1.
item	Код товара согласно каталога компании.
filterValue	Filter Value кода для указания типа упаковки, для которой предназначен данный EPC.
serial	Серийный номер экземпляра товара, необязательный параметр.

Пример 1. Товар для продажи на кассе, серийные номера ведутся клиентом самостоятельно:

```
// Создание EPC на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «4», что означает условно «Наша компания» и, соответственно,
// сгенерированный EPC будет «нашим внутренним EPC», как, например, штрихкоды EAN13 вида «20.....»
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы,
                               СерийныйНомер);
// или (то же самое)
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, 0, СерийныйНомер);
// для компании, зарегестрированной в Юнискан
epc = КлеверенсРФИД.EPCизSGTIN(КодКомпании_в_Юнискан, КодТовара, 0, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерируются «Wonderfid™ Link».

```
// Создание EPC на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «4», что означает условно «Наша компания» и, соответственно,
// сгенерированный EPC будет «нашим внутренним EPC», как, например, штрихкоды EAN13 вида «20.....»
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// или (то же самое)
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара);
// для компании, зарегестрированной в Юнискан
epc = КлеверенсРФИД.EPCизSGTIN(КодКомпании_в_Юнискан, КодТовара);
```

Генерирование EPC для палет и коробок

Пример 1. Маркировка палеты для внутреннего использования (не выходит за рамки склада):

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
// Код компании указан как «4», что означает условно «Наша компания».
epc = КлеверенсРФИД.EPCизSSCC(4, ЧисловойНомерПаллеты);
```

Пример 2. Маркировка палеты для внешнего использования (выходит за рамки склада):

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
// Код компании должен быть получен при регистрации в Юнискан (GS1).
epc = КлеверенсРФИД.EPCизSSCC(КодКомпании_в_Юнискан, ЧисловойНомерПаллеты);
```

Пример 3. Маркировка поддона/ коробки/ пробирки как оборачивамой тары:

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
epc = КлеверенсРФИД.EPCизGRAI(КодКомпании_в_Юнискан, ТипТары, СерийныйНомер);
// например:
epc = КлеверенсРФИД.EPCизGRAI(КодКомпании_в_Юнискан, ТипТары, СерийныйНомер);
// Код компании указан как «4», что означает условно «Наша компания».
epc = КлеверенсРФИД.EPCизGRAI(4, ТипТары, СерийныйНомер);
```

Генерирование EPC для документов

Пример 1. Маркировка документа накладной:

```
// Создание EPC для накладной.  Маркируются сквозным уникальным числовым номером.
epc = КлеверенсРФИД.EPCизGDTI(КодКомпании_в_Юнискан, НомерТипаДокумента, СерийныйНомер);
// например:
epc = КлеверенсРФИД.EPCизGDTI(4062146, 04021, 44200122213);
// Код компании указан как «4», что означает условно «Наша компания».
epc = КлеверенсРФИД.EPCизGDTI(4, ТипТары, СерийныйНомер);
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Для библиотечных задач

Последние изменения: 2024-03-26

> Более подробно о применении «Wonderfid™ Link» для библиотечных задач читайте в разделе «UHF RFID для библиотек».

Стандарт ISO 28560 RFID в библиотеках предусматривает RFID-учет всех библиотечных объектов. С помощью RFID в рамках стандарта можно учитывать:

1. библиотечный фонд – книги, журналы, диски и т.п., выдаваемые абонементам;
2. читательские билеты (метка либо вклеивается в билет, либо сам билет заменяется RFID-карточкой);
3. собственное библиотечное имущество, не выдаваемое абонементам (столы, шкафы и т.п.);
4. товары на продажу;
5. списанные объекты и объекты, ожидающие утилизации.

«Wonderfid™ Link» поддерживает всё из вышеперечисленного.

Цифровое кодирование UII

В RFID-метку UII записывается при помощи нулей и единиц. Перевод UII в нули и единицы называется бинарным кодированием UII, которое уже реализовано в продукте (самим ничего кодировать и декодировать не нужно). Самый распространенный способ записи UII – это строка, представляющая собой последовательную запись в 16-ричном формате всех байт бинарно закодированного UII, и именно в таком виде UII отображают программы, которые идут с RFID-оборудованием по умолчанию.

Строки вида «3024000003320C4063A23312», которые выдаются демопрограммой считывателя как TAG ID считанных RFID-меток, требуют декодирования в EPC или UII. Только тогда можно узнать номер библиотечного объекта, на который нанесена метка, и т.п. Все стандартные схемы кодирования/декодирования уже реализованы в «Wonderfid™ Link», ничего самим писать не нужно.

Пример декодирования UII при помощи «Wonderfid™ Link»:

```
uii = КлеверенсРФИД.ДекодироватьEPCUII("608940C09996D7A00000");
```

Пример кодирования UII при помощи «Wonderfid™ Link»:

```
строка = КлеверенсРФИД.UIIизБиблиотечногоКода(Неопределено, "10054123").БинарноеПредставление;
```

Общий алгоритм маркировки

Поскольку метки прошиваются конкретным библиотечным кодом, все их следует прошивать по очереди. Наиболее удобный способ – сначала оптом обклеить интересующие объекты «непрошитыми» метками, а затем по одному прошить уникальным кодом.

Пример алгоритма маркировки:

```
// получить с сервера используемый пароль на доступ к RFID-меткам
парольНаДоступ = ПолучитьПарольНаДоступRFID();
Пока Истина Цикл
    // Заставить пользователя выбрать из базы конкретный объект фонда, читательский билет и т.п.
    // если выбранному объекту уже сопоставлена метка – переспросить пользователя
    // (например, метка могла выйти из строя и действительно требуется перемаркировка)
    маркируемыйОбъект = ВыбратьЭкземпляр();
    Если маркируемыйОбъект = Неопределено Тогда
        Возврат;
    КонецЕсли;

    режим = РежимДиалогаВопрос.ОКОтмена;
    выбраннаяМетка = Неопределено;
    Пока выбраннаяМетка = Неопределено Цикл
        ответ = Неопределено;
        метки = Неопределено;
        // Поискать вокруг антенны RFID-метки в течение 1й секунды (1000 миллисекунд)
        Попытка
            // Читаем не только Tag ID, но и банк TID меток, чтобы потом писать в конкретную.
            метки = считыватель.ПрочестьМетки (5000, Истина, Ложь, Ложь);
        Исключение
            Вопрос("Ошибка поиска меток! " + КлеверенсРФИД.ОписаниеОшибки(), РежимДиалогаВопрос.ОК);
            Продолжить;
        КонецПопытки;

        Если метки.Количество = 0 Тогда
            ответ = Вопрос("Положите маркируемый объект на антенну!", режим);
        Иначе
            Если метки.Количество > 0 Тогда
                ответ = Вопрос("Уберите от антенны посторонние предметы!", режим);
            Иначе
                // Выбрать единственную метку
                выбраннаяМетка = метки.Элемент(0);
            КонецЕсли;
        КонецЕсли;

        Если ответ = КодВозвратаДиалога.Отмена Тогда
            Прервать;
        КонецЕсли;
    КонецЦикла;

    Попытка
        // Создать UII в соответствии с тем, какой объект выбрали:
        uii = СоздатьПравильныйUII(маркируемыйОбъект);
        // Записать UII
        считыватель.ЗаписатьEPCUIIпоTID(выбраннаяМетка.TagId, выбраннаяМетка.TID, uii, парольНаДоступ);

        Сообщить("В метку с Tag ID [" + выбраннаяМетка.TagId + "] успешно записан новый UII [" +
                                    uii.Строка() + "] (" + uii.БинарноеПредставление + ").");
    Исключение
        Предупреждение("Ошибка записи в метку! " + КлеверенсРФИД.ОписаниеОшибки());
    КонецПопытки;
КонецЦикла;
```

Маркировка библиотечного фонда

Пример №1. если у библиотеки нет ISIL

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(Неопределено, экземпляр.Код);
```

Пример №2. если у библиотеки есть ISIL:

```
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, экземпляр.Код);
```

Пример №3. если не жалко памяти RFID-метки:

```
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, экземпляр.Код);
// если память метки позволяет, то можно проставить в UII тип использования для объекта
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.ДляВыдачи;
```

UII следует записать в банк EPCUII.

Если в метке очень много памяти, то в банк USER можно записать дополнительную информацию об объекте фонда.

Пример №4. Прошивка в банк USER наименования книги, номера тома и места на полке:

```
// получить с сервера используемый пароль на доступ к RFID-меткам
парольНаДоступ = ПолучитьПарольНаДоступRFID();

бо = КлеверенсРФИД.СоздатьБиблиотечныйОбъект();
бо.Наименование = "Л. Н. Толстой.  Война и Мир, том 1й";
бо.РазмерНабора = 4;  // 4 тома
бо.НомерВНаборе = 1;  // 1й том
бо.МестоНаПолке = "А-14-21";

банк = бо.СформироватьUSERБанк();
// заполненные выше данные займут ровно 74 байта памяти банка USER
// метки с банком памяти USER < 74 байа не смогут быть прошитыми
считыватель.ЗаписатьUSER(метка.TagId, банк, парольНаДоступ);
```

Данные из приведенного примера займут ровно 74 байта банка памяти USER. Самые бюджетные метки в настоящий момент имеют всего 32 бита памяти USER и, соответственно, не смогут быть использованы в таком сценарии.

Маркировка читательских билетов (и RFID-карточек)

Пример №1. Формирование UII для читательского билета:

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, читатель.Код, КлеверенсРФИД.AFI.Библиотечный);
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет;
```

Пример №2. Проверка, что тип использования у метки – любой читательский билет:

```
Если метка.Объект.Тип() = "БиблиотечныйКод" И метка.Объект.ТипИспользования <> Неопределено И
метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.ЧитательскийБилет.КодКласса
Тогда
```

Если требуется по логике и позволяет память метки, то можно прошить в банк USER некие дополнительные данные о держателе билета.

Пример №3. Прошивка в банк USER имени владельца билета:

```
// получить с сервера используемый пароль на доступ к RFID-меткам
парольНаДоступ = ПолучитьПарольНаДоступRFID();

бо = КлеверенсРФИД.СоздатьБиблиотечныйОбъект();
бо.Наименование = читатель.ФИО;

банк = бо.СформироватьUSERБанк();
считыватель.ЗаписатьUSER(метка.TagId, банк, парольНаДоступ);
```

Маркировка библиотечного имущества (столы и стулья)

Пример №1. Формирование UII для библиотечного имущества:

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uii = КлеверенсРФИД.UIIизБиблиотечногоКода(ISIL, имущество.Код, КлеверенсРФИД.AFI.НаСкладе);
uii.ТипИспользования = КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество;
```

Пример №2. Проверка, что тип использования у метки – любое имущество:

```
Если метка.Объект.Тип() = "БиблиотечныйКод" И метка.Объект.ТипИспользования <> Неопределено И
(метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.Имущество.КодКласса или
метка.Объект.ТипИспользования.КодКласса = КлеверенсРФИД.Библиотеки.ТипыИспользования.НеДляВыдачи.КодКласса) Тогда
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Виртуальный режим

Последние изменения: 2024-03-26

Для тестирования работы компоненты без RFID-считывателя на руках, в ней предусмотрен так называемый «виртуальный режим», в котором компонента подключается к виртуальным считывателям и читает виртуальные метки. «Виртуальный» в данном случае означает «отсутствующий на самом деле».

Для активации виртуального режима используется следующий код:

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.Включен = Истина;
```

Настройки виртуального режима позволяют задать параметры работы несуществующих считывателей так, чтобы они удовлетворяли условиям проводимых тестов.

Пример №1 | виртуальное чтение всегда ровно 6-ти случайных меток.

В такой настройке компонента сгенерирует шесть случайных меток и будет их «читать».

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 6;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 0;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

Пример №2 | виртуальное чтение от 6-ти до 10-ти (раз на раз не приходится) случайных меток.

В такой настройке компонента будет от инвентаризации к инвентаризации генерировать от шести до десяти случайных меток.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 6;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 10;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

Пример №3 | виртуальное чтение двух заранее заданных меток.

В такой настройке компонента всегда будет «читать» только две указанные метки.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 2;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 2;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000001");
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000002");
```

Пример №4 | виртуальное чтение двух заранее заданных и одной-двух случайных меток.

В такой настройке компонента от инвентаризации к инвентаризации будет «читать» либо две указанные метки + одна случайная, либо две указанные + две случайных.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 3;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 4;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000001");
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("300800000000000000000002");
```

Пример №5 | виртуальное чтение трех заранее заданных и нескольких случайных меток.

В такой настройке компонента от инвентаризации к инвентаризации будет генерировать от ноля до семи случайных меток и «читать» их наряду с тремя заранее заданными.

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 3;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 10;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Очистить();

// создаем метку по Tag ID.
tagid1 = "300800000000000000000000";
метка1 = КлеверенсРФИД.НоваяМетка(tagid1);
// Аттрибут «Счетчик» означает число меток с идентичным EPC.  Если Счетчик = 2, то при инвентаризации были
// обнаружены две метки с идентичным EPC.  В реальной инвентаризации вместо того, чтобы возвратить две
// одинаковые метки, компонента объединяет их в одну, и проставляет счетчик = 2.
метка1.Счетчик = 2;
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка1);

// создаем EPC единицы товара с серийным номером «4412»,  кодом товара «123» от фирмы с кодом «7770».
// первый ноль означает, что это EPC товара для продажи на кассе.
epc = КлеверенсРФИД.EPCизSGTIN(0, 7770, 123, "4412");
// создаем метку по EPC.
метка2 = КлеверенсРФИД.НоваяМетка(epc);
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка2);

// создаем EPC единицы товара с серийным номером «332», сам товар задаем по EAN13.
// первый ноль означает, что это EPC товара для продажи на кассе.
epc = КлеверенсРФИД.EPCизEAN13(0, "4004764390793", "332");
// создаем метку по EPC.
метка3 = КлеверенсРФИД.НоваяМетка(epc);
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка3);
```

Пример №6 | сначала какое-то время виртуально читается только одна метка, затем только другая.

В некоторых ситуациях для тестирования алгоритмов учета может понадобиться управлять сценарием считывания меток. Например, чтобы сразу после запуска инвентаризации читались какие-то одни определенные метки, а спустя пару секунд – другие определенные метки. В приведенной ниже настройке от инвентаризации к инвентаризации компонента будет воспроизводить один и тот же сценарий: сначала «читается» метка "300800000000000000000001", затем она исчезает и начинает «читаться» метка "300800000000000000000002".

любой модуль:

```
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМин = 2;
КлеверенсРФИД.ВиртуальныйРежим.ЧислоМетокМакс = 2;

ТестовыеМетки = КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки;

// добавляем метку по Tag ID.  Метка начинает читаться спутся примерно 1 сек. и видна примерно 5 сек.
ТестовыеМетки.ДобавитьПоВремени("300800000000000000000001", 1, 5);
// добавляем метку по Tag ID.  Метка начинает читаться на 8й сек. и видна примерно 2 сек.
ТестовыеМетки.ДобавитьПоВремени("300800000000000000000002", 8, 2);
```

Не нашли что искали?

Задать вопрос в техническую поддержку

Чтение меток

Последние изменения: 2024-03-26

Операция инвентаризации поддерживается на уровне радио-протокола обмена между метками и считывателем, и возвращает данные о том, какие EPC присутствуют в зоне считывания.

Например, все метки могут иметь один и тот же EPC/UII, и в этом случае по итогам инвентаризации мы будем знать, что это за EPC, и сколько всего RFID-меток с этим EPC/UII удалось считать ридеру.

Если все метки имеют свой уникальный EPC/UII (не путать с уникальным номером чипа, который безусловно есть у каждой метки Class 1 Gen 2), то операция инвентаризации вернет список этих EPC/UII.

Синхронная инвентаризация (чтение) меток

Синхронная инвентаризация означает следующее:

1. «1С:Предприятие» дало считывателю команду «считай окружающие метки в течение N секунд» и замерло в ожидании ответа.
2. Считыватель читает метки, «1С:Предприятие» ждет, все формочки замерли. Считыватель закончил через указанное время и вернул результат «1С:Предприятию».
3. «1С:Предприятие» получило результат, осознала его, формочки «отвисли».

① ② ③

Таким образом, если при синхронной инвентаризации указать считывателю «считай 50 секунд», то окно 1С почти целую минуту не будет доступно для пользователя.

Пример кода для синхронной инвентаризации:

Модуль формы:

```
// ------------------------ по нажатии кнопки 1 ---------------------------
// Опрашивать окружающие метки в течение 5000 миллисекунд (5 сек)
метки = считыватель.ПрочитатьМетки(5000);
Для индекс = 0 по метки.Количество - 1 Цикл
        метка = метки.Элемент(индекс);
        ОбработатьМетку(метка); // Какая-то процедура обработки метки
КонецЦикла;
```

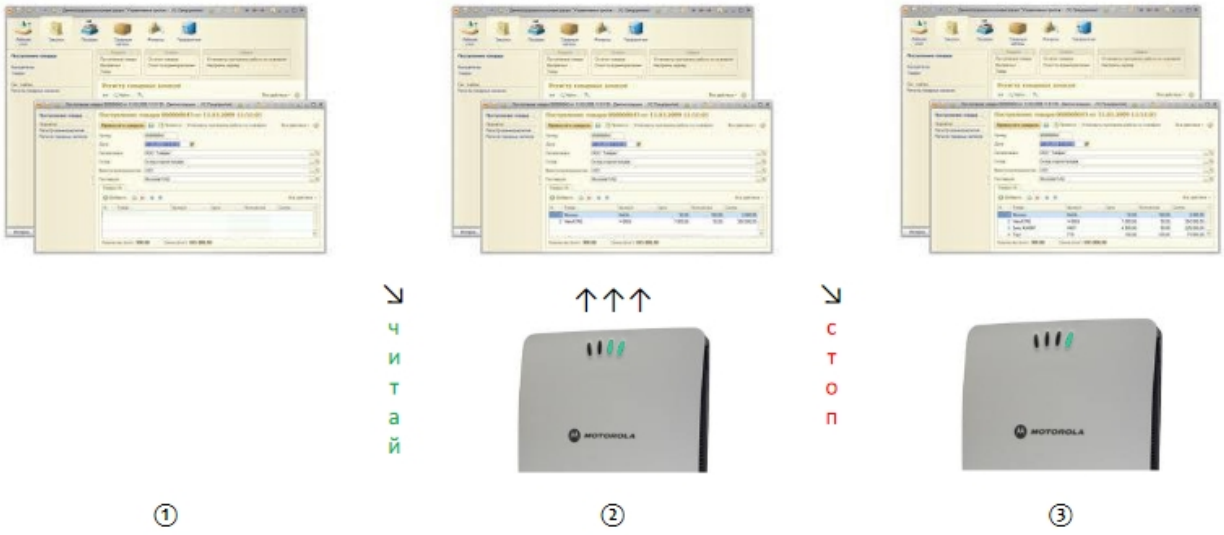
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

Во время синхронной инвентаризации внешнее событие «Чтение» не приходит, т.к. это «убило» бы приложение 1С.

Асинхронная инвентаризация (чтение) меток

Асинхронная инвентаризация означает следующее:

1. «1С:Предприятие» дало считывателю команду «считай окружающие метки в течение N секунд» и продолжило делать свои дела.
2. По мере инвентаризации новых меток считыватель асинхронно посылает «1С:Предприятию» внешние события, в результате чего считанные метки могут интерактивно появляться в окнах и документах «1С:Предприятия».
3. Считыватель либо закончил через указанное время, либо «1С:Предприятие» дало ему команду закончить инвентаризацию досрочно.



Таким образом, при асинхронной инвентаризации окно 1С всегда остается доступным для взаимодействия с пользователем, а найденные метки могут интерактивно появляться на экране.

Пример кода для асинхронной инвентаризации:

```
// Опрашивать окружающие метки в течение 5000 миллисекунд (5 сек)
считыватель.НачатьЧтение(5000);

// Получить все метки, обнаруженные во время инвентаризации (включая и те, по которым приходили события)
метки = считыватель.ОкончитьЧтение();
Для индекс = 0 по метки.Количество - 1 Цикл
    метка = метки.Элемент(индекс);
    ОбработатьМетку(метка);
КонецЦикла;
```

Событие «Чтение»

При каждом удачном асинхронном чтении RFID-метки (в частности, при асинхронной инвентаризации) посылается внешнее событие «Чтение».

Источник = "CleverenceRFID"

Событие = "Чтение"

Данные = Tag ID прочитанной метки, например «303000181CE257587E9CA77C».

Более подробную информацию о самой метке можно получить у конкретного считывателя или у самой компоненты через метод «ВыбратьМетку».

В качестве данных в событие приходит только Tag ID метки. Получить более подробные данные можно при помощи метода компоненты «ВыбратьМетку».

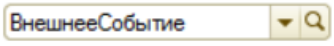
Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "Чтение" Тогда
    Попытка
        // Получить полные данные считанной метки (или одинаковых меток) сразу со всех считывателей:
        метка = КлеверенсРФИД.ВыбратьМетку(Данные);
        // Либо получить данные у конкретного считывателя (подробнее о считывателях см. ниже)
        // метка = считыватель.ВыбратьМетку(tagid);

        Сообщить(метка.TagId + ", кол-во: " + метка.Счетчик +
                    " шт., время='" + метка.Время.Строка() + "', RSSI=" + метка.RSSI);
        ...
    Исключение
        Сообщить(КлеверенсРФИД.ОписаниеОшибки());
    ОкончаниеПопытки;
  КонецЕсли;
КонецПроцедуры
```

либо, если подписать форму на событие «ВнешнееСобытие»:

ВнешнееСобытие 

Модуль формы:

```
Процедура ВнешнееСобытие(Источник, Событие, Данные)
  Если Источник = "CleverenceRFID" И Событие = "Чтение" Тогда
    Попытка
      // Работа с компонентой
      // Получить полные данные считанной метки (или одинаковых меток) сразу со всех считывателей:
      метка = КлеверенсРФИД.ВыбратьМетку(Данные);
      // Либо получить данные у конкретного считывателя (подробнее о считывателях см. ниже)
      // метка = считыватель.ВыбратьМетку(tagid);

      Сообщить(метка.TagId + ", кол-во: " + метка.Счетчик +
                  " шт., время='" + метка.Время.Строка() + "', RSSI=" + метка.RSSI);
      ...
    Исключение
      Сообщить(КлеверенсРФИД.ОписаниеОшибки());
    ОкончаниеПопытки;
  КонецЕсли;
КонецПроцедуры
```

Чтение банка EPC/UII

Чтение банка EPC/UII происходит во время инвентаризации меток (которая не требует паролей), а также при чтении любых других банков, поэтому отдельно чтением банка EPC/UII озадачиваться необязательно.

Чтение банка USER

Банк USER хранит любую дополнительную информацию в формате ISO 15961 (конкретные упакованные поля со строковыми значениями) либо просто байтами. В зависимости от используемого в метке чипа, банк USER может быть размером от ноля бит до нескольких килобайт.

Пример №1:

Любой модуль:

```
// Прочесть банки USER всех меток в поле видимости считывателя, в течение 2,5 секунд (2500 миллисекунд)
метки = считыватель.ПрочестьБанкUSER(2500);
Для индекс = 0 по метки.Количество - 1 Цикл
        метка = метки.Элемент(индекс);
        Сообщить("Прочитано: " + метка.tagID + ", USER = " + Строка(метка.БанкUSER));
КонецЦикла;
```

Пример №2:

Любой модуль:

```
// Прочесть банк USER у первой же метки, Tag ID которой равен указанному.
банкTID = считыватель.ПрочестьБанкUSER("3024000003320C4063A23312");
Сообщить("Прочитано: USER = " + метка.БанкUSER.Строка());
```

Чтение банка TID (запись запрещена)

Банк TID хранит уникальный номер чипа. Перепрошить этот номер чипа никак нельзя. Если при маркировке объектов вести реестр всех использованных чипов, то банк TID можно использовать для проверки того, что метка не была «заменена злоумышленником».

Пример №1:

Любой модуль:

```
// Прочесть банки TID всех меток в поле видимости считывателя, в течение 1,5 секунд (1500 миллисекунд)
// пароль на доступ = 0 (нет пароля).
метки = считыватель.ПрочестьБанкTID(1500, 0);
Для индекс = 0 по метки.Количество - 1 Цикл
        метка = метки.Элемент(индекс);
        Сообщить("Прочитано: " + метка.tagID +
                                        ", MDID = " + метка.БанкTID.MDID + ", TMN = " + метка.БанкTID.TMN);
КонецЦикла;
```

Пример №2:

Любой модуль:

```
// Прочесть банк TID у первой же метки, Tag ID которой равен указанному.  Пароль на доступ = 0 (нет пароля).
банкTID = считыватель.ПрочестьБанкTID("3024000003320C4063A23312", 0);
Сообщить("Прочитано: MDID = " + банкTID.MDID + ", TMN = " + банкTID.TMN);
```

Чтение и запись банка RESERVED

Банк RESERVED хранит пароли на доступ и блокирование метки. Если метки используются только внутри организации и никуда не передаются, то в целях защиты от несанкционированного перепрошивания меток сторонними лицами всегда имеет смысл установить единый секретный пароль хотя бы на доступ к чтению/записи.

Поскольку на чтение банка RESERVED нужно знать пароль доступа, то большого смысла в операции чтения содержимого банка RESERVED ради пароля доступа нет. Однако, некоторые производители включают в банк RESERVED дополнительную информацию, например альтернативный пароль доступа, с которым читается второй «приватный» набор банков (что позволяет организовать «публичную» и «внутреннюю» версии данных одной и той же метки).

Пример:

Любой модуль:

```
// Прочесть банк RESERVED у первой же метки, Tag ID которой равен указанному. Пароль на доступ = 123.
банкRESERVED = считыватель.ПрочестьБанкRESERVED("3024000003320C4063A23312", 123);
Сообщить("Прочитано: пароль доступа = " + банкRESERVED.ПарольДоступа +
          ", пароль на блокирование = " + банкRESERVED.ПарольНаБлокирование);
дополнительныеПароли = банкRESERVED.ДополнительныеБайты;
```

Задать вопрос в техническую поддержку

Запись банка RFID-меток

Последние изменения: 2024-03-26

Операция записи банка поддерживается на уровне радио-протокола обмена между метками и считывателем и позволяет переписать всю или часть информации в интересующем банке RFID-меток (если эту память не прожгли намертво). В рамках одного запроса можно писать в любое количество банков и любое количество меток одновременно. Считыватель отправляет запрос, а метки, подходящие под условия запроса, каждая по очереди записывается.

Запись сразу в несколько меток

Из 4х банков меток Gen2 для записи доступны три: банк с паролями, банк EPC и пользовательский банк.

Текущая реализация компоненты такова, что записать что-либо в метку можно только зная её Tag ID (чтобы не писать непонятно что в случайные метки). Поэтому прежде чем что-нибудь записать, сначала следует проинвентаризовать метки и получить их Tag ID.

Зная Tag ID, можно записать что-нибудь одновременно во все метки с таким Tag ID.

Любой модуль:

```
Попытка
        // Создать EPC:
        epc = ...

        // Записать EPC:
        ПодключенныйСчитыватель.ЗаписатьEPCUII(ИнтересуемаяМетка.TagId, epc, 0);
        Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "] успешно записан новый EPC [" +
                                        epc.Строка() + "] (" + epc.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
                                                        КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Запись только в одну конкретную метку

Запись только в одну конкретную метку опирается на то, что у каждой метки должен быть свой уникальный номер чипа.

Зная EPC, можно прочитать банк TID одой единственной метки и затем записать только в неё:

Любой модуль:

```
Попытка
        новыйEPC = ...

        // Прочесть номер чипа, пароль на чтение = 0 (нет пароля):
        tid = ПодключенныйСчитыватель.ПрочестьБанкTID(ИзвестныйTagID, 0);
        // Записать новый EPC по номеру чипа, пароль на запись = 0 (нет пароля):
        ПодключенныйСчитыватель.ЗаписатьEPCпоTID(ИзвестныйTagID, tid, новыйEPC, 0);
        Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "] успешно записан новый EPC [" +
                                        новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
                                                        КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Не зная TID можно просто прочесть банки TID всех меток вокруг и мотом записать в нужную:

Любой модуль:

```
Попытка
        новыйEPC = ...

        // Читать метки и банки TID всех меток вокруг в течение 1,5 сек (1500 миллисекунд)
        // пароль на чтение = 0 (нет пароля)
        // возвратится коллекция меток, в каждой из которых будет проставлен реквизит TID
        метки = ПодключенныйСчитыватель.ПрочестьБанкиTID(1500, 0);
        // Записать новый EPC по номеру чипа, пароль на запись = 0 (нет пароля):
        ПодключенныйСчитыватель.ЗаписатьEPCпоTID(метка[0].TagId, метка[0].TID, новыйEPC, 0);
        Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "]  успешно записан новый EPC [" +
                                        новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
        Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
                                                                КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Событие «Запись»

При каждой удачной асинхронной записи RFID-метки компонента посылает внешнее событие «Запись».

Источник = "CleverenceRFID"

Событие = "Запись"

Данные = Tag ID записываемой метки, например «303000181CE257587E9CA77C» (старый Tag ID, т.к. после записи в банк EPC Tag ID метки мог поменяться).

Более подробная информация недоступна, метод «ВыбратьМетку» не применим.

В качестве данных в событие приходит только Tag ID метки. Получить более подробные данные можно при помощи метода компоненты «ВыбратьМетку», который принимает Tag ID и возвращает объект компоненты с описанием метки (см. «Событие «Чтение»).

Пример кода обработки события:

Модуль управляемого приложения:

```
Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные) // Предопределенная процедура 1С
  // Глобальный обработчик внешнего события
  Если Источник = "CleverenceRFID" И Событие = "Запись" Тогда
    // Сообщить Tag ID записанной метки:
    Сообщить("Записана метка: " + Данные);
  КонецЕсли;
КонецПроцедуры
```

Для логистики и розницы

Последние изменения: 2024-03-26

Генерирование EPC для товаров

Если метки используются для целей контроля за движением товаров/объектов/документов, то самым главным в RFID-метке будет являться банк EPC. В банке EPC/UII будет содержаться собственно EPC или UII, описывающий, на какой конкретно объект будет нанесена RFID-метка.

Под генерированием EPC понимаются правила, по которым компания будет заполнять поля EPC перед их записью в метку. Данные для заполнения берутся либо из 1С, либо прямо из штрихкодов товаров. Эти правила следует выработать для каждого типа маркируемых объектов, чтобы правильно настроить работу RFID-принтера и/или выделенного маркировочного места со стационарным RFID-считывателем.

EPC по штрихкоду товара

«Wonderfid™ Link» предоставляет несколько способов создания EPC на основе штрихкодов товаров. В примерах ниже ШК – любой штрихкод EAN8, EAN13, ISBN, ISSN или UPC.

Пример 1. Товар для продажи на кассе, серийные номера генерирует 1С:

```
// EPC товара на основе штрихкода и уникального серийного номера единицы товара
epc = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерирует сам продукт:

```
// EPC товара только на основе штрихкода (уникальный серийный номер будет сгенерирован
// компонентой при записи в метку, см. в разделе «Ошибка! Источник ссылки не найден.»)
epc = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// либо (то же самое)
epc = КлеверенсРФИД.EPCизEAN13(ШК);
```

EPC по коду товара

«Wonderfid™ Link» предоставляет несколько способов создания EPC по коду товара.

```
/// <example>
/// Предположим, штрихкод товара равен "2209537497279", а серийный номер изделия - "207".
/// В этом штрихкоде "2209537497" - код компании, "27" - код товара, "9" - чексумма.
/// Тогда создание соответствующего EPC будет выглядеть следующим образом:
/// <br/>
/// Пример использования библиотеки из Visual Basic 6:
/// <code lang="VBScript">
/// Set epc = api.EPCfromSGTIN(0, 2209537497, 27, "9")
/// </code>
/// Пример использования библиотеки из «1С:Предприятие 8»:
/// <code lang="1C 8">
/// epc = КлеверенсРФИД.EPCизSGTIN(0, 2209537497, 27, "9");
/// </code>
/// </example>
```

Пример 1. Товар для продажи на кассе, серийные номера ведутся клиентом самостоятельно:

```
// Создание EPC на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «4», что означает условно «Наша компания» и, соответственно,
// сгенерированный EPC будет «нашим внутренним EPC», как, например, штрихкоды EAN13 вида «20.....»
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы,
                               СерийныйНомер);
// или (то же самое)
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, 0, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерируются самим продуктом:

```
// Создание EPC на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «4», что означает условно «Наша компания» и, соответственно,
// сгенерированный EPC будет «нашим внутренним EPC», как, например, штрихкоды EAN13 вида «20.....»
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// или (то же самое)
epc = КлеверенсРФИД.EPCизSGTIN(4, КодТовара);
```

Генерирование EPC для документов

«Wonderfid™ Link» предоставляет много способов создания EPC на основе данных о товарах, упаковках, основных средствах или документах:

Любой модуль:

```
Попытка
	// Создание EPC на основе штрихкода EAN13 и уникального серийного номера единицы товара
	epc = КлеверенсРФИД.EPCизEAN13(
			КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы.Значение, EAN13, СерийныйНомер);

	// Создание EPC на основе штрихкода EAN13 и уникального серийного номера паллеты с товаром
	epc = КлеверенсРФИД.EPCизEAN13(
			КлеверенсРФИД.ФильтрыEPC.SGTIN_Контейнер.Значение, EAN13, СерийныйНомер);

	// Создание EPC на основе кода товара и уникального серийного номера единицы товара
	// Код компании указан как «2», что означает условно «Наша компания» и, соответственно,
	// сгенерированный EPC будет «нашим внутренним EPC», как, например, штрихкоды EAN13 вида «20.....»
	epc = КлеверенсРФИД.EPCизSGTIN(
			КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы.Значение,
			2, НоменклатураКод, СерийныйНомер);

	// Создание EPC для паллеты с товаром.  Паллеты маркируются сквозным уникальным номером.
	// Код компании указан как «2», что означает условно «Наша компания»...
	epc = КлеверенсРФИД.EPCизSSCC(
			КлеверенсРФИД.ФильтрыEPC.SSCC_Все.Значение, 2, ЧисловойНомерПаллеты);


	// Создание EPC на основе числового кода типа документа и номера конкретного документа.
	// Код компании указан как «2», что означает условно «Наша компания» и, соответственно,
	// сгенерированный EPC будет «нашим внутренним EPC»
	epc = КлеверенсРФИД.EPCизGDTI(
			КлеверенсРФИД.ФильтрыEPC.GDTI_Все.Значение,
			2, ЧисловойТипДокумента, СерийныйНомерДокумента);

Исключение
	Предупреждение("Ошибка создания EPC: " + КлеверенсРФИД.ОписаниеОшибки());
КонецПопытки;
```

Генерирование EPC для маркировки палет и коробок

Пример 1. Маркировка палеты для внутреннего использования (не выходит за рамки склада):

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
// Код компании указан как «4», что означает условно «Наша компания».
epc = КлеверенсРФИД.EPCизSSCC(4, ЧисловойНомерПаллеты);
```

Пример 2. Маркировка палеты для внутреннего использования (выходит за рамки склада):

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
// Код компании должен быть получен при регистрации в Юнискан (GS1).
epc = КлеверенсРФИД.EPCизSSCC(КодКомпании_в_Юнискан, ЧисловойНомерПаллеты);
```

Пример 3. Маркировка палеты для публичной циркуляции:

```
// Создание EPC для паллеты или коробки.  Маркируются сквозным уникальным числовым номером.
// Код компании указан как «4», что означает условно «Наша компания».
```

Не нашли что искали?

Задать вопрос в техническую поддержку