

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:

```
КлеверенсRFID.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:

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМин = 6;
```

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМакс = 0;
```

```
КлеверенсRFID.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:

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМин = 6;
```

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМакс = 10;
```

```
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

Пример №3 | виртуальное чтение двух заранее заданных меток.

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

любой модуль:

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМин = 2;
```

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМакс = 2;
```

```
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Очистить();
```

```
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Добавить("30080000000000000000000001");
```

```
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Добавить("30080000000000000000000002");
```

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("30080000000000000000000000000001");
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить("30080000000000000000000000000002");
```

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

В такой настройке компонента от инвентаризации к инвентаризации будет генерировать от нуля 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 = "30080000000000000000000000000000";
метка1 = КлеверенсРФИД.НоваяМетка(tagid1);
// Атрибут «Счетчик» означает число меток с идентичным ЕРС. Если Счетчик = 2, то при инвентаризации были
// обнаружены две метки с идентичным ЕРС. 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 ЕРС единицы товара с серийным номером «4412», кодом товара «123» от фирмы с кодом «7770».
// первый ноль означает, что это ЕРС товара для продажи на кассе.
ерс = КлеверенсРФИД.ЕРСизSGTIN(0, 7770, 123, "4412");
// создаем метку по ЕРС.
метка2 = КлеверенсРФИД.НоваяМетка(ерс);
КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки.Добавить(метка2);

// создаем ЕРС единицы товара с серийным номером «332», сам товар задаем по EAN13.
// первый ноль означает, что это ЕРС товара для продажи на кассе.
ерс = КлеверенсРФИД.ЕРСизEAN13(0, "4004764390793", "332");
// создаем метку по ЕРС.
метка3 = КлеверенсРФИД.НоваяМетка(ерс);
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 "30080000000000000000000000000001", затем она исчезает и начинает «читаться» метка "30080000000000000000000000000002".

любой модуль:

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

ТестовыеМетки = КлеверенсРФИД.ВиртуальныйРежим.ТестовыеМетки;

// добавляем метку по Tag ID. Метка начинает читаться спустя примерно 1 сек. и видна примерно 5 сек.
ТестовыеМетки.ДобавитьПоВремени("30080000000000000000000000000001", 1, 5);
// добавляем метку по Tag ID. Метка начинает читаться на 8й сек. и видна примерно 2 сек.
ТестовыеМетки.ДобавитьПоВремени("30080000000000000000000000000002", 8, 2);
```

Не нашли что искали?



Задать вопрос в техническую поддержку

Обработка внешних событий в «1С:Предприятия»

Последние изменения: 2024-03-26

По мере работы компоненты в predetermined procedure «ОбработкаВнешнегоСобытия» основного модуля «1С:Предприятия», а также в procedure «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 работает).

По физическому подключению и настройке RFID-считывателей [«Установка и настройка RFID считывателей»](#).

Синхронный поиск считывателей

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

Пример кода синхронного поиска считывателей:

любой модуль:

```

считыватели = КлеверенсRFID.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:

```
КлеверенсRFID.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 predeterminedенную 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 uI 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 uI найденного RFID-считывателя:
    Сообщить("Найден считыватель: " + Данные);
КонецЕсли;
КонецПроцедуры

```

Не нашли что искали?



Задать вопрос в техническую поддержку

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

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-ричным 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 UUI) прочитанной метки.	
НомерАнтенны	
Antennald	
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/UUI). 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](#)

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».



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;
```

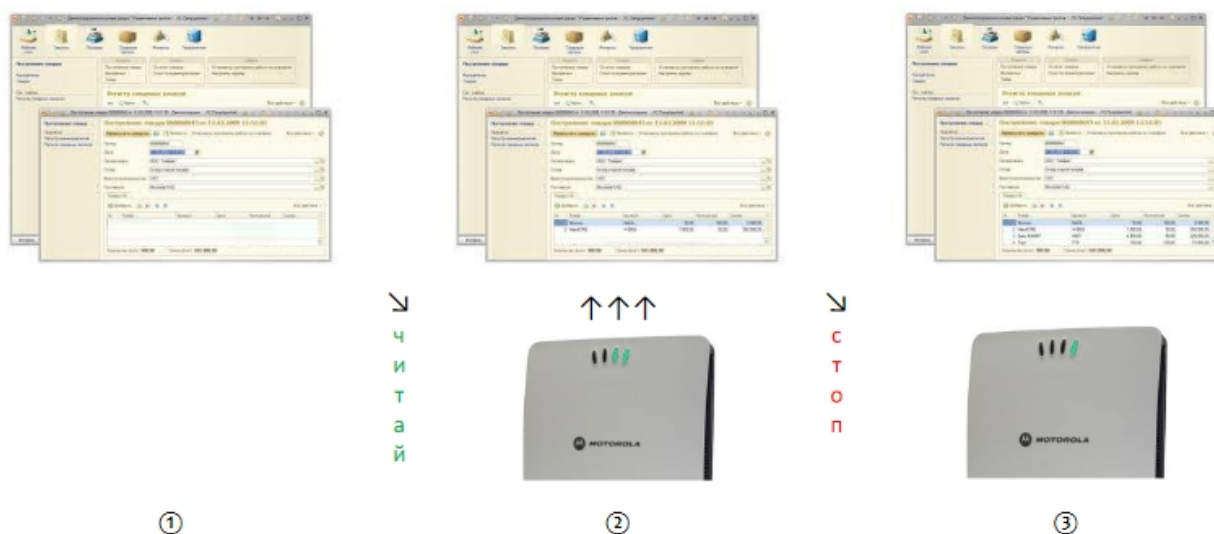
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

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 = КлеверенсRFID.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(КлеверенсRFID.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 = КлеверенсRFID.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(КлеверенсRFID.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:

```

// Прочсть банки USER всех меток в поле видимости считывателя, в течение 2,5 секунд (2500 миллисекунд)
метки = считыватель.ПрочстьБанкUSER(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:

```
// Прочсть банк USER у первой же метки, Tag ID которой равен указанному.
банкTID = считыватель.ПрочстьБанкUSER("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:

```
// Прочсть банки TID всех меток в поле видимости считывателя, в течение 1,5 секунд (1500 миллисекунд)
// пароль на доступ = 0 (нет пароля).
метки = считыватель.ПрочстьБанкTID(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:

```
// Прочсть банк TID у первой же метки, Tag ID которой равен указанному. Пароль на доступ = 0 (нет пароля).
банкTID = считыватель.ПрочстьБанкTID("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:

```
// Прочсть банк RESERVED у первой же метки, Tag ID которой равен указанному. Пароль на доступ = 123.
банкRESERVED = считыватель.ПрочстьБанкRESERVED("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:
```

```
ерс = ...
```

```
// Записать EPC:
```

```
ПодключенныйСчитыватель.ЗаписатьEPCUII(ИнтересуемаяМетка.TagId, ерс, 0);
```

```
Предупреждение("В метку с tag ID [" + ИнтересуемаяМетка.TagId + "] успешно записан новый EPC [" + ерс.Строка() + "] (" + ерс.БинарноеПредставление + ").");
```

Исключение

```
Предупреждение("Ошибка записи в метку [" + ИнтересуемаяМетка.TagId + "]: " +
```

```
КлеверенсRFID.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 одной единственной метки и затем записать только в неё:

Любой модуль:

Попытка

```
новыйEPC = ...
```

```
// Прочсть номер чипа, пароль на чтение = 0 (нет пароля):
```

```
tid = ПодключенныйСчитыватель.ПрочстьБанкTID(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 + "]: " +
```

```
КлеверенсRFID.ОписаниеОшибки());
```

КонецПопытки;

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.ПрочитатьБанкиTID(1500, 0);

// Записать новый EPC по номеру чипа, пароль на запись = 0 (нет пароля):

ПодключенныйСчитыватель.ЗаписатьEPCпоTID(метка[o].TagId, метка[o].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 + "]: " +

КлеверенсRFID.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

Не нашли что искали?



Задать вопрос в техническую поддержку

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
ерс = 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.»)
ерс = КлеверенсРФИД.EPCизEAN13(ШК, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// либо (то же самое)
ерс = 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">
/// ерс = КлеверенсРФИД.ЕРСизSGTIN(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.....»
ерс = КлеверенсРФИД.ЕРСизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы,
                               СерийныйНомер);

// или (то же самое)
ерс = КлеверенсРФИД.ЕРСизSGTIN(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.....»
ерс = КлеверенсРФИД.ЕРСизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыEPC.SGTIN_ТоварДляКассы);
// или (то же самое)
ерс = КлеверенсРФИД.ЕРСизSGTIN(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 ЕРС на основе штрихкода EAN13 и уникального серийного номера единицы товара
ерс = КлеверенсРФИД.ЕРСизEAN13(
    КлеверенсРФИД.ФильтрыЕРС.SGTIN_ТоварДляКассы.Значение, EAN13, СерийныйНомер);

// Создание ЕРС на основе штрихкода EAN13 и уникального серийного номера паллеты с товаром
ерс = КлеверенсРФИД.ЕРСизEAN13(
    КлеверенсРФИД.ФильтрыЕРС.SGTIN_Контейнер.Значение, EAN13, СерийныйНомер);

// Создание ЕРС на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «2», что означает условно «Наша компания» и, соответственно,
// сгенерированный ЕРС будет «нашим внутренним ЕРС», как, например, штрихкоды EAN13 вида «20.....»
ерс = КлеверенсРФИД.ЕРСизSGTIN(
    КлеверенсРФИД.ФильтрыЕРС.SGTIN_ТоварДляКассы.Значение,
    2, НоменклатураКод, СерийныйНомер);

// Создание ЕРС для паллеты с товаром. Паллеты маркируются сквозным уникальным номером.
// Код компании указан как «2», что означает условно «Наша компания»...
ерс = КлеверенсРФИД.ЕРСизSSCC(
    КлеверенсРФИД.ФильтрыЕРС.SSCC_Все.Значение, 2, ЧисловойНомерПаллеты);

// Создание ЕРС на основе числового кода типа документа и номера конкретного документа.
// Код компании указан как «2», что означает условно «Наша компания» и, соответственно,
// сгенерированный ЕРС будет «нашим внутренним ЕРС»
ерс = КлеверенсРФИД.ЕРСизGDTI(
    КлеверенсРФИД.ФильтрыЕРС.GDTI_Все.Значение,
    2, ЧисловойТипДокумента, СерийныйНомерДокумента);
```

Исключение

```
Предупреждение("Ошибка создания ЕРС: " + КлеверенсРФИД.ОписаниеОшибки());
```

КонецПопытки;

Генерирование ЕРС для маркировки палет и коробок

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

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

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

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

Пример 3. Маркировка палеты для публичной циркуляции:

```
// Создание ЕРС 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