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;
```

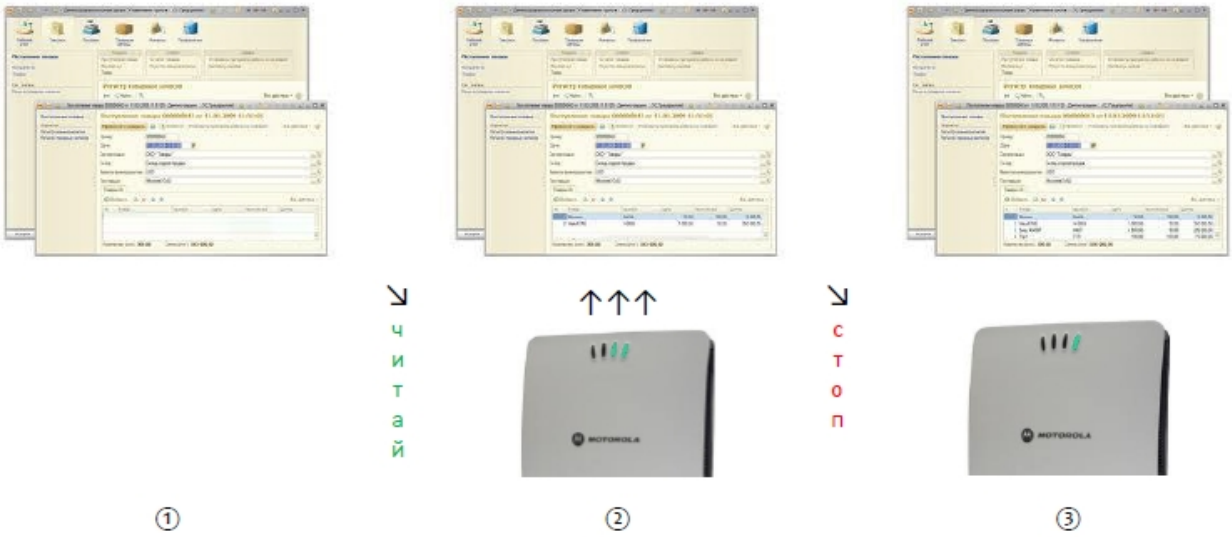
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

> 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