

Документация по компоненте Cleverence RFID для стационарных считывателей

Июнь 2016 (версия 1.3.0.2)

Примеры кода работы с компонентой для «1С:Предприятия»

Motorola FX9500



**Motorola
FX7400**



Motorola XR480



Impinj Speedway



**RoyalRay
RRU9809USBL**



NordicID Sampo S1



Содержание

Введение	2
Начало работы с компонентой	2
Завершение работы с компонентой.....	2
Обработка ошибок при работе с компонентой.....	3
Распространенные ошибки и их устранение.....	3
Виртуальный режим работы компоненты.....	4
Внешние события компоненты	6
Поиск и подключение RFID-считывателей	7
Синхронный поиск считывателей.....	7
Асинхронный поиск считывателей.....	7
Событие «НайденСчитыватель»	7
RFID-метка – объект компоненты	10
Чтение меток	11
Синхронное чтение (инвентаризация) меток.....	11
Асинхронное чтение (инвентаризация) меток.....	12
Событие «Чтение».....	13
Событие «ЧтениеОкончено»	14
Чтение банка EPC/UII	15
Чтение банка USER	15
Чтение банка TID (запись в него невозможна).....	15
Чтение банка RESERVED	16
Запись меток.....	17
Запись сразу в несколько меток	17
Запись только в одну конкретную метку.....	17
Событие «Запись»	18
Для задач логистики и розницы.....	19
Цифровое кодирование EPC	19
Генерирование EPC для товаров	19
Серийные номера EPC для товаров	20
EPC по штрихкоду товара	20
EPC по коду товара	21
Генерирование EPC для паллет и коробок	21
Генерирование EPC для документов.....	22
Для библиотечных задач.....	23
Цифровое кодирование UII.....	23
Общий алгоритм маркировки.....	24
Маркировка библиотечного фонда.....	25
Маркировка читательских билетов (и RFID-карточек).....	25
Маркировка библиотечного имущества (столы и стулья)	26

Введение

Полная документация по API и объектам компоненты (все поля, процедуры и функции) входит в дистрибутив компоненты под названием «Клеверенс RFID. Часть 10. Справочник разработчика» и ставится вместе с компонентой. Ищите в папке установки КОМПОНЕНТЫ.

Начало работы с компонентой

Для начала работы с компонентой следует создать **один** её экземпляр на каждом клиенте (который будет работать с RFID-считывателями со своей локальной машины):

Модуль управляемого приложения:

```
&НаКлиенте
Перем КлеверенсRFID Экспорт;

Процедура ПриНачалеРаботыСистемы()
  Попытка
    ПодключитьВнешнююКомпоненту("AddIn.CleverenceRFID");
    КлеверенсRFID = новый СОМОбъект("AddIn.CleverenceRFID");
  Исключение
    Предупреждение("Не удалось подключить компоненту AddIn.CleverenceRFID. Возможно, она не
зарегистрирована в системе", 20, "Ошибка подключения компоненты");
  Возврат;
КонецПопытки;

КлеверенсRFID.Язык = "Русский";
КонецПроцедуры
```

Завершение работы с компонентой

Процедура правильного завершения работы с компонентой (**при закрытии окна 1С** или остановке сервера) должна выглядеть следующим образом:

Модуль управляемого приложения:

```
КлеверенсRFID.ЗавершитьРаботуСКомпонентой();
КлеверенсRFID = Неопределено;
```

Обработка ошибок при работе с компонентой

Если в работе компоненты возникают ошибки (исключительные ситуации), то «1С:Предприятие» в основном интерпретирует их как «Неизвестная ошибка». Чтобы получить более подробную информацию по ошибке, следует использовать метод «ОписаниеОшибки()» компоненты:

любой модуль:

```
Попытка
    // Работа с компонентой
    ...
Исключение
    Если ЗначениеЗаполнено(КлеверенсRFID.ОписаниеОшибки()) Тогда
        ОписаниеОшибки = КлеверенсRFID.ОписаниеОшибки();
    Иначе
        ОписаниеОшибки = ОписаниеОшибки();
    КонецЕсли;
    Сообщить(ОписаниеОшибки);
ОкончаниеПопытки;
```

Кроме того все ошибки, возникающие при работе самой компоненты (т.е. те, которые возникли после успешного выполнения функции «ПодключитьВнешнююКомпоненту»), пишутся в лог-файл «CleverenceRFID_log.txt», который лежит в папке «Мои документы» того пользователя, под которым сейчас запущено «1С:Предприятие». Некоторые редкие ошибки выполнения функции «ПодключитьВнешнююКомпоненту» тоже пишутся в этот лог.

Распространенные ошибки и их устранение

В работе компоненты выявлены ошибки, которые появляются либо на некоторых конфигурациях машин, либо время от времени на любых конфигурациях.

При любой непонятной ошибке первым делом не лишне будет посмотреть файл [CleverenceRFID_log.txt](#), который лежит в папке «Мои документы» того пользователя, под которым сейчас запущено «1С:Предприятие»!

Метод «ОбработкаВнешнегоСобытия» в 1С не вызывается

Убедитесь, что ваш компьютер удовлетворяет требованиям, перечисленным в разделе «**Ошибка! Источник ссылки не найден.**» части документации по установке и настройке компоненты. Посмотрите, может быть что-то написано в [CleverenceRFID_log.txt](#)

Чтение меток асинхронно (асинхронная инвентаризация) ничего не читает

Подробнее см. разделы про предварительную настройку RFID считывателей и «**Ошибка! Источник ссылки не найден.**» части документации по установке и настройке компоненты.

«Приложение будет закрыто» и обрушение окна «1С:Предприятия»

Возникает при повторных созданиях и уничтожениях компоненты, например при повторных открытиях копий демонстрационной обработки в одном и том же окне «1С:Предприятия». Совет: создавайте не более одного экземпляра компоненты на каждого запущенного клиента 1С.

Виртуальный режим работы компоненты

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("3008000000000000000001");  
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Добавить("3008000000000000000002");
```

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:

```
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМин = 3;  
КлеверенсRFID.ВиртуальныйРежим.ЧислоМетокМакс = 4;  
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Очистить();  
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Добавить("3008000000000000000001");  
КлеверенсRFID.ВиртуальныйРежим.ТестовыеМетки.Добавить("3008000000000000000002");
```

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 = "30080000000000000000";
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 "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 спустя 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);
```

Внешние события компоненты

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 рассказано ниже в описании конкретных сценариев работы компоненты.

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 см. «**Ошибка! Источник ссылки не найден.**».

Синхронный поиск считывателей

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

Примечание: во время синхронного поиска событие «НайденСчитыватель» не приходит, т.к. такое событие «подвесило» бы 1С.

Пример кода синхронного поиска считывателей (перед этим необходимо начать работу с компонентой, см «Начало работы с компонентой»):

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» (см. «Внешние события компоненты»
 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:

5. НайденСчитыватель;
6. Чтение;
7. ЧтениеОкончено;
8. Запись.

Более подробно о каждом из событий рассказано ниже в описании конкретных сценариев работы компоненты. Поиск и подключение RFID-считывателей»).

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»:

ВнешнееСобытие	<input type="text" value="ВнешнееСобытие"/>
Модуль формы:	
Процедура	ВнешнееСобытие(Источник, Событие, Данные)
	Если Источник = "CleverenceRFID" И Событие = "НайденСчитыватель" Тогда

```
// Сообщить u1 найденного RFID-считывателя:  
Сообщить("Найден считыватель: " + Данные);  
КонецЕсли;  
КонецПроцедуры
```

RFID-метка – объект компоненты

В рамках компоненты RFID-метка представляет собой специальный объект с набором реквизитов и функций. Наиболее подробно все объекты компоненты рассмотрены в документе «Клеверенс RFID. Часть 10. 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 UII) 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. <div style="border: 1px solid gray; padding: 5px; margin-top: 5px;">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.</div>
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.

Чтение меток

Операция инвентаризации поддерживается на уровне радио-протокола обмена между метками и считывателем и возвращает **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, 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 нажатию кнопки 1 -----  
// Опрашивать окружающие метки в течение 5000 миллисекунд (5 сек)  
метки = считыватель.ПрочитатьМетки (5000);  
Для индекса = 0 по метки.Количество - 1 Цикл  
метка = метки.Элемент(индекс);  
ОбработатьМетку(метка); // Какая-то процедура обработки метки  
КонецЦикла;
```

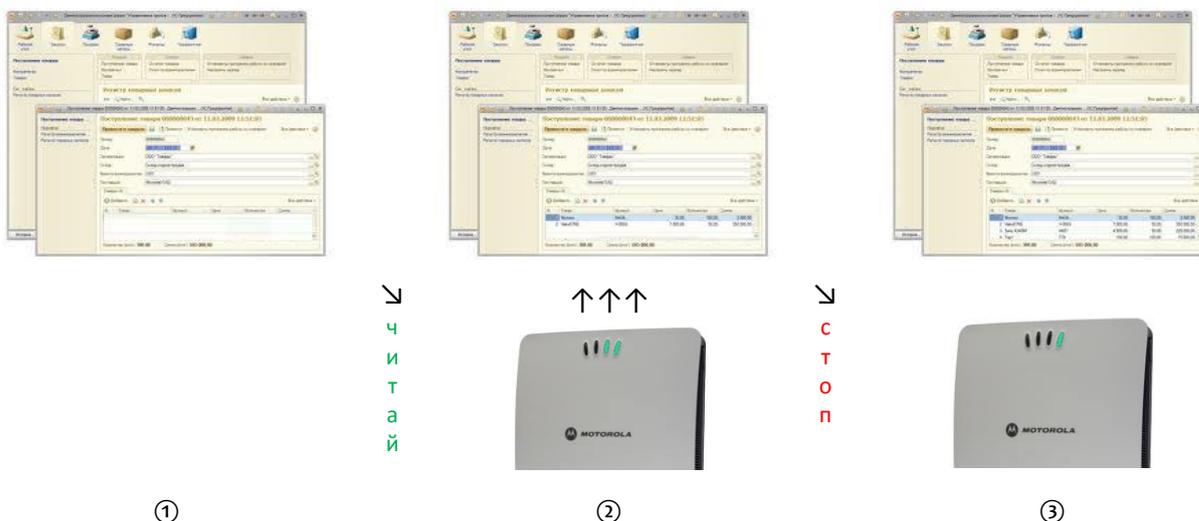
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

Примечание: 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 чтения 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 (создается методом НачатьЧтение), ugi 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 = 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

```
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
```

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

```
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 ноля бит до нескольких килобайт.

Пример №1:

Любой модуль:

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

```
// Прочеть банк USER у первой же метки, Tag ID которой равен указанному.
банкTID = считыватель.ПрочетьБанкUSER("302400003320C4063A23312");
Сообщить("Прочитано: USER = " + метка.БанкUSER.Строка());
```

Чтение банка TID (запись в него невозможна)

Банк TID хранит уникальный номер чипа. Переписать 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:

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

```
// Прочеть банк TID у первой же метки, Tag ID которой равен указанному. Пароль на доступ = 0 (нет пароля).
банкTID = считыватель.ПрочетьБанкTID("302400003320C4063A23312", 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:

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

Запись меток

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:
```

```
епс = ...
```

```
// 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 + "]: " +
```

```
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.

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.ПрочстьМетки(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 + "]: " +
        КлеверенсRFID.ОписаниеОшибки());
КонецПопытки;

```

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 [" + ИзвестныйTagID + "] успешно записан новый EPC [" +
        новыйEPC.Строка() + "] (" + новыйEPC.БинарноеПредставление + ").");
Исключение
    Предупреждение("Ошибка записи в метку [" + ИзвестныйTagID + "]: " +
        КлеверенсRFID.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

```

Для задач логистики и розницы

Более подробно о том, что такое EPC и зачем он нужен, см. документ «Клеверенс RFID. Часть 5. UHF RFID для логистики и розницы».

Цифровое кодирование EPC

В RFID-метку EPC записывается при помощи нулей и единиц. Перевод EPC в ноли и единицы называется бинарным кодированием EPC, которое уже реализовано в компоненте (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 «3024000003320C4063A23312», 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 компоненте Клеверенс, ничего самим писать не нужно.

Пример декодирования EPC при помощи компоненты Клеверенс:

```
епс = КлеверенсRFID.ДекодироватьEPCUII("3024000003320C4063A23312");
```

Пример кодирования EPC при помощи компоненты Клеверенс:

```
строка = КлеверенсRFID.EPCизSGTIN(о, 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 (см. «**Ошибка! Источник ссылки не найден.**»), 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 нолём). 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, следует читать в документе «Клеверенс RFID. Часть 5. 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, компонента 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 компонентой Клеверенс 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 **Неопределено**, то в созданном EPC он будет пустым и, соответственно, компонента сгенерирует его перед записью в метку.

EPC по штрихкоду товара

Компонента «Клеверенс RFID» 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 = КлеверенсRFID.EPCизEAN13(ШК, КлеверенсRFID.ФильтрыEPC.SGTIN_ТоварДляКассы, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерирует компонента Клеверенс:

```
// 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 = КлеверенсRFID.EPCизEAN13(ШК, КлеверенсRFID.ФильтрыEPC.SGTIN_ТоварДляКассы, Неопределено);  
// либо (то же самое)  
ерс = КлеверенсRFID.EPCизEAN13(ШК, КлеверенсRFID.ФильтрыEPC.SGTIN_ТоварДляКассы);  
// либо (то же самое)  
ерс = КлеверенсRFID.EPCизEAN13(ШК);
```

ЕРС по коду товара

API метод компоненты ЕРСизSGTIN (ЕРСfromSGTIN) создает экземпляр SGTIN-варианта ЕРС 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 ЕРС.
serial	Серийный номер экземпляра товара, необязательный параметр.

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

```
// Создание ЕРС на основе кода товара и уникального серийного номера единицы товара
// Код компании указан как «4», что означает условно «Наша компания» и, соответственно,
// сгенерированный ЕРС будет «нашим внутренним ЕРС», как, например, штрихкоды EAN13 вида «20.....»
ерс = КлеверенсРФИД.ЕРСизSGTIN(4, КодТовара, КлеверенсРФИД.ФильтрыЕРС.SGTIN_ТоварДляКассы,
                               СерийныйНомер);

// или (то же самое)
ерс = КлеверенсРФИД.ЕРСизSGTIN(4, КодТовара, 0, СерийныйНомер);
// для компании, зарегистрированной в Юнискан
ерс = КлеверенсРФИД.ЕРСизSGTIN(КодКомпании_в_Юнискан, КодТовара, 0, СерийныйНомер);
```

Пример 2. Товар для продажи на кассе, серийные номера генерируются компонентой Клеверенс:

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

Генерирование ЕРС для паллет и коробок

Пример 1. Маркировка паллеты 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 паллеты для внешнего 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 поддона/коробки/пробирки как оборачиваемой тары:

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

Генерирование ЕРС для документов

Пример 1. Маркировка документа накладной:

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

Для библиотечных задач

Более подробно о применении компоненты Клеверенс для библиотечных задач см. документ «Клеверенс RFID. Часть 6. 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 клеивается 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.

Компонента Клеверенс поддерживает всё из вышеперечисленного.

Цифровое кодирование UII

В RFID-метку UII записываются при помощи нулей и единиц. Перевод UII в ноли и единицы называется бинарным кодированием UII, которое уже реализовано в компоненте (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 компоненте Клеверенс, ничего самим писать не нужно.

Пример декодирования UII при помощи компоненты Клеверенс:

```
uII = КлеверенсRFID.ДекодироватьEPCUII("608940C09996D7A00000");
```

Пример кодирования UII при помощи компоненты Клеверенс:

```
строка = КлеверенсRFID.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.ПрочстьМетки (5000, Истина, Ложь, Ложь);
        Исключение
            Вопрос("Ошибка поиска меток! " + КлеверенсRFID.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! " + КлеверенсRFID.ОписаниеОшибки());
    КонецПопытки;
КонецЦикла;
```

Маркировка библиотечного фонда

Пример №1. если у библиотеки нет ISIL

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uіі = КлеверенсРФІД.УІІизБиблиотечногоКода(Неопределено, экземпляр.Код);
```

Пример №2. если у библиотеки есть ISIL:

```
uіі = КлеверенсРФІД.УІІизБиблиотечногоКода(ISIL, экземпляр.Код);
```

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

```
uіі = КлеверенсРФІД.УІІизБиблиотечногоКода(ISIL, экземпляр.Код);
// если память метки позволяет, то можно проставить в UІІ тип использования для объекта
uіі.ТипИспользования = КлеверенсРФІД.Библиотеки.ТипыИспользования.ДляВыдачи;
```

UІІ следует записать в банк EPCUІІ.

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 = КлеверенсРФІД.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 байта 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 UІІ для читательского билета:

```
// если у библиотеки нет ISIL, то можно передать Неопределено
uіі = КлеверенсРФІД.УІІизБиблиотечногоКода(ISIL, читатель.Код, КлеверенсРФІД.АFІ.Библиотечный);
uіі.ТипИспользования = КлеверенсРФІД.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 = КлеверенсРФІД.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 = КлеверенсRFID.СоздатьБиблиотечныйОбъект();  
бо.Наименование = читатель.ФИО;
```

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

Маркировка библиотечного имущества (столы и стулья)

Пример №1. Формирование UИИ для библиотечного имущества:

```
// если у библиотеки нет ISIL, то можно передать Неопределено  
uИИ = КлеверенсRFID.UИИизБиблиотечногоКода(ISIL, имущество.Код, КлеверенсRFID.AFI.НаСкладе);  
uИИ.ТипИспользования = КлеверенсRFID.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 = КлеверенсRFID.Библиотеки.ТипыИспользования.Имущество.КодКласса или  
метка.Объект.ТипИспользования.КодКласса = КлеверенсRFID.Библиотеки.ТипыИспользования.НеДляВыдачи.КодКласса) Тогда
```