Чтение RFID меток

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, 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.

- 1 Синхронное чтение (инвентаризация) меток
- 2 Асинхронное чтение (инвентаризация) меток
- 3 Событие «Чтение»
- 4 Событие «ЧтениеОкончено»
- 5 Чтение банка EPC/UII
- 6 Чтение банка USER
- 7 Чтение банка TID (запись в него невозможна)
- 8 Чтение банка RESERVED

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;
```

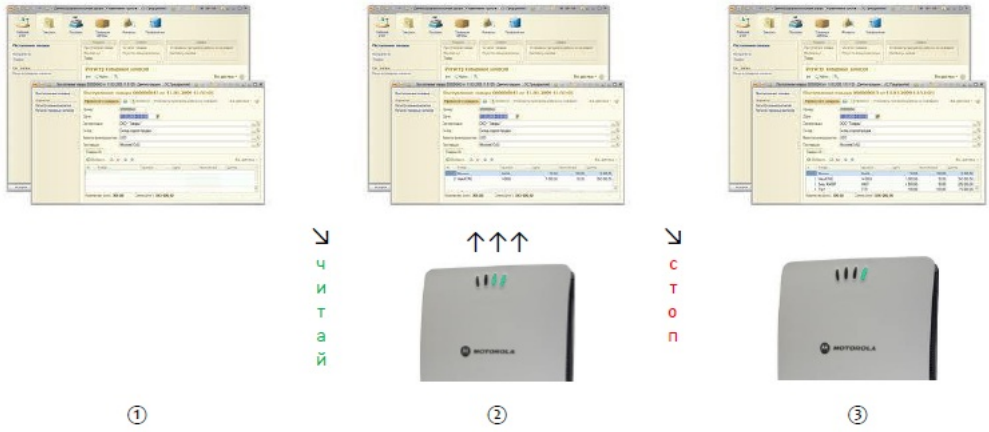
Синхронная инвентаризация не требует обрабатывания внешнего события «Чтение», и поэтому работает во всех конфигурациях «1С:Предприятия 8.2» и всех версиях операционной системы Windows.

> 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 дел
2. По мере инвентаризации новых меток считыватель асинхронно посылает «1С:Предприятию» внешние события, в результате чег 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 URI 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»:

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("3024000003320C4063A23312");
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("3024000003320C4063A23312", 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("3024000003320C4063A23312", 123);
Сообщить("Прочитано: пароль доступа = " + банкRESERVED.ПарольДоступа +
        ", пароль на блокирование = " + банкRESERVED.ПарольНаБлокирование);
дополнительныеПароли = банкRESERVED.ДополнительныеБайты;
```

Была ли статья полезна?

Нет

Да