



**SECUROS  
POS**

Версия 10

**Руководство по интеграции  
POS-терминалов**

Руководство по интеграции POS-терминалов SecurOS POS (POSINTEG - RU, сборка 122 от 24.04.2020).

© Copyright Intelligent Security Systems, 2020.

Отпечатано в России.

Intelligent Security Systems оставляет за собой право вносить изменения как в данное Руководство, так и в описываемый продукт. Изменения могут вноситься в спецификацию системы без уведомления. Содержимое Руководства не является офертой, гарантией, обещанием или условием договора, и не должно восприниматься подобным образом.

Никакая часть данной документации не может быть воспроизведена, передана, процитирована, размещена в поисковой системе, переведена на любой язык или машинный код в любой форме и любыми средствами без явного письменного согласия со стороны правообладателя. Несанкционированное копирование этой публикации может не только нарушить авторские права, но и ослабить возможность Intelligent Security Systems предоставлять точную и актуальную информацию пользователям продукта.

# Содержание

<b>1 Предисловие</b>	<b>4</b>
1.1 Назначение	4
1.2 Целевая аудитория	4
1.3 Использование руководства	4
1.4 Обращение за технической поддержкой	4
1.5 Соглашение по оформлению	6
1.6 Элементы оформления	6
<b>2 Общее описание</b>	<b>7</b>
<b>3 Процедура интеграции</b>	<b>8</b>
3.1 Создание XML-файла описателя парсера	8
3.2 Реализация JS-парсера	9
3.2.1 Методы объекта listener	9
3.2.1.1 pushReceiptBegin	10
3.2.1.2 pushCashierName	10
3.2.1.3 pushNewSale	10
3.2.1.4 pushSaleAmount	10
3.2.1.5 pushSaleDiscount	10
3.2.1.6 pushSaleProductsCount	11
3.2.1.7 pushSaleArticle	11
3.2.1.8 pushSaleEnd	11
3.2.1.9 pushReceiptDiscount	11
3.2.1.10 pushReceiptEnd	11
3.2.1.11 pushFilteredString	12
3.2.1.12 pushEvent	12
3.2.1.13 setEventReceiptCode	13
3.2.1.14 setReceiptCode	13
<b>4 Пример интеграции</b>	<b>14</b>
<b>5 Приложения</b>	<b>19</b>
5.1 Приложение 1. Информация для Службы технической поддержки	19
<b>Предметный указатель</b>	<b>21</b>

# 1 Предисловие

Данный раздел содержит общую информацию о текущем документе, средствах его оформления и использования, а также о порядке получения дополнительной поддержки при эксплуатации продукта.

## 1.1 Назначение

Данное руководство описывает процесс интеграции кассовых терминалов в систему безопасности SecurOS для их совместной работы с модулем SecurOS POS.

## 1.2 Целевая аудитория

Руководство предназначено для системных интеграторов и администраторов SecurOS — опытных пользователей компьютера, имеющих навыки по установке аппаратного обеспечения и построению сети на основе протокола TCP/IP, знающих основы технологий CCTV.

## 1.3 Использование руководства

Данный документ можно использовать как в печатном, так и в электронном виде. В последнем случае доступны такие возможности ПО Adobe Reader, как закладки и гипертекстовые ссылки для навигации по документу. Данное руководство ссылается на другие документы по SecurOS (**Краткое руководство пользователя SecurOS** и т. д.), которые можно найти на установочном диске SecurOS или на веб-сайте компании ISS ([www.iss.ru](http://www.iss.ru)).

## 1.4 Обращение за технической поддержкой

При наличии вопросов, ответы на которые отсутствуют в данном руководстве, обратитесь к администратору системы или системному интегратору.

За дальнейшей информацией обращайтесь в Службу технической поддержки компании Intelligent Security Systems:

---

Примечание. Для оперативной реакции на запрос используйте Портал технической поддержки, [www-адрес которого указан ниже](http://www.iss.ru).

---

- **в США:**

по телефону: +1 732 855 1111 (с понедельника по пятницу с 8:30 до 18:00 EST);

e-mail: [support@issivs.com](mailto:support@issivs.com)

www: <https://support.issivs.com>

- **в России:**

по телефону: +7 (495) 645 21 21 (многоканальный, с понедельника по четверг с 9:00 до 18:00, в пятницу с 9:00 до 17:00 по московскому времени);

www: <https://help.iss.ru>

---

Примечание. Инструкция по работе с Порталом размещена на <https://help.iss.ru/user/manual>.

---

- **в Бразилии:**

по телефону: +55 11 2262 2894 (с понедельника по пятницу с 9:00 до 18:00 BRT);

e-mail: [suporte@issivs.com](mailto:suporte@issivs.com)

www: <https://support.issivs.com>

- **в Мексике:**

по телефону: +52 1 551330 0181 (с понедельника по пятницу с 9:00 до 18:00 CDT);

e-mail: [supportlatam@issivs.com](mailto:supportlatam@issivs.com)

www: <https://support.issivs.com>

- **в Колумбии/Эквадоре:**

по телефону: +57 300 442 2808 (с понедельника по пятницу с 9:00 до 18:00 COT/ECT);

e-mail: [supportlatam@issivs.com](mailto:supportlatam@issivs.com)

www: <https://support.issivs.com>

- **в Чили:**

по телефону: +56 9 6573 2993 (с понедельника по пятницу с 9:00 до 18:00 CLT);

e-mail: [supportlatam@issivs.com](mailto:supportlatam@issivs.com)

www: <https://support.issivs.com>

- **в Украине:**

по телефону: +380 (44) 299 08 10 (с понедельника по пятницу с 9:00 до 18:00 EET);

e-mail: [supportua@issivs.com](mailto:supportua@issivs.com)

www: <https://support.issivs.com>

- **в Перу/Боливии:**

по телефону: +51 997 111 678 (с понедельника по пятницу с 9:00 до 18:00 PET/BOT);

e-mail: [supportlatam@issivs.com](mailto:supportlatam@issivs.com)

www: <https://support.issivs.com>

- **в Аргентине:**

по телефону: +54 91152528779 (с понедельника по пятницу с 9:00 до 18:00 ART);

e-mail: [supportlatam@issivs.com](mailto:supportlatam@issivs.com)

www: <https://support.issivs.com>

Для более скорого разрешения проблем, рекомендуем подготовить служебную информацию, указанную в разделе [Информация для Службы технической поддержки](#).

## 1.5 Соглашение по оформлению

В данном документе для представления различных терминов и названий используются следующие шрифты и средства форматирования.

Параметр	Описание
<b>жирный</b>	Используется при написании названий рабочих мест, утилит или экранных форм, окон и диалоговых окон, а также названий их элементов.
<i>курсив</i>	Используется для выделения объектов SecurOS.
<b><i>жирный курсив</i></b>	Используется для выделения элементов однородных списков.
моноширинный	Используется для выделения текстов макрокоманд и программных кодов, имен файлов и путей к ним. Также используется для указания необходимой опции, выделения значений, задаваемых пользователем с клавиатуры.
<b>зеленый</b>	Используется для выделения перекрестных ссылок внутри документа и ссылок на доступные внешние документы.

## 1.6 Элементы оформления

**Внимание!** Служит для привлечения внимания пользователя к информации, которая необходима для корректного восприятия изложенного далее текста. Как правило, данная информация имеет предупреждающий характер.

**Примечание.** Текст примечания в основном тексте.

### **Дополнительная информация**

Используется для отображения информации дополнительного характера. В элементах такого типа размещается, например, описание вариантов выполнения операции или ссылка на дополнительную литературу.

## 2 Общее описание

Интеграция кассовых терминалов с модулем SecurOS POS системы SecurOS дает пользователю дополнительные возможности по управлению и мониторингу системы, а также возможность совместного использования различных кассовых терминалов с другими подсистемами (например, видео- и аудиоконтроля, см. [Руководство администратора SecurOS](#)).

Чтобы интегрировать кассовый терминал в систему SecurOS выполните следующие действия:

1. Установите ПО Интеграционной платформы видеоменеджмента SecurOS.
2. Установите ПО модуля SecurOS POS. Создайте и настройте необходимые объекты модуля и объекты системы SecurOS. По вопросам конфигурации системы и установки (подключения) кассовых терминалов рекомендуем обратиться к [Руководству пользователя SecurOS POS](#).
3. Создайте шаблоны для составления скрипта синтаксического анализатора (парсера) чеков (см. [Процедура интеграции](#)).
4. Запустите SecurOS.
5. Настройте объект *POS Терминал*. В окне настройки параметров объекта в поле **Профиль разборщика** выберите имя созданного файла парсера (см. рис. 1).

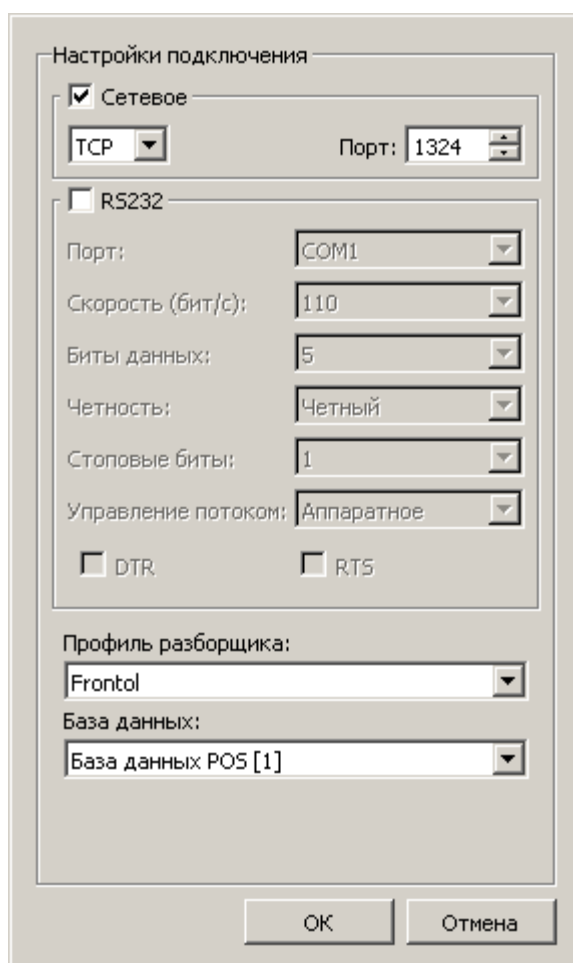


Рис. 1. Окно настройки параметров объекта POS Терминал

## 3 Процедура интеграции

Механизм интеграции кассовых терминалов основан на запуске скрипта JavaScript (JS-парсер), составленного с использованием библиотеки интеграции JavaScript.

Процедура интеграции кассовых чеков состоит из следующих этапов:

1. **Создание XML-файла описателя парсера.**
2. **Реализация JS-парсера.**

### 3.1 Создание XML-файла описателя парсера

В подкаталоге \POSParsers корневой папки SecurOS находятся XML-описатели парсеров, специфичные для каждого типа кассового терминала, и JS-файлы, которые реализуют операции в этих парсерах. XML-файл описателя парсера имеет следующий вид:

**Листинг 1.** Пример XML-описателя парсера

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>Test profile</comment>
  <entry key="profile_name">POSParsers/test.js</entry>
  <entry key="charsetname">cp866</entry>
  <entry key="classname">ru.iss.pos.server.parser.js.
  ReceiptProcessorJSImpl</entry>
</properties>
```

---

**Примечание.** Допустимые способы модификации XML-описателя описаны в DTD-документе, который опубликован на сайте <http://java.sun.com/dtd/properties.dtd>.

---

Чтобы создать XML-описатель парсера выполните следующие действия:

1. Скопируйте один из XML-описателей парсера в файл с названием, соответствующим новому парсеру.
2. Модифицируйте файл XML-описателя нового парсера:
  - В поле атрибута `profile_name` пропишите путь к JS-файлу, относительный к корневому каталогу SecurOS.
  - В поле атрибута `charsetname` укажите тип кодировки данных, поступающих с кассового терминала (через COM- или UDP-порт). Имя кодировки должно соответствовать стандарту Java. Список поддерживаемых кодировок см. на <http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html>. Например, для случая кассовых терминалов, передающих данные в DOS-кодировке, значение `charsetname` должно быть установлено как `cp866` (см. Листинг 1 выше). Если атрибут `charsetname` не установлен, то никаких операций по перекодировке данных не делается. См. также документацию по обработке потоков в языке Java: <http://java.sun.com/j2se/1.4.2/docs/api/java/io/InputStreamReader.html>.



**Внимание!** Для кассовых терминалов, передающих данные через UDP-порт, атрибут `charsetname` не используется.

## 3.2 Реализация JS-парсера

---

**Примечание.** Скрипт парсера должен быть реализован в соответствии со стандартом языка JavaScript v 1.5. Для получения подробной справочной информации см. [http://developer.mozilla.org/en/docs/Core\\_JavaScript\\_1.5\\_Reference](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference).

---

JS-скрипт синтаксического анализатора (парсера) чеков предназначен для анализа и извлечения данных из кассового чека, поступающего с терминала.

Чтобы создать JS-скрипт парсера выполните следующие действия:

1. Скопируйте файл `dummy.js` из подкаталога `SecurOS\POSParsers` в каталог, указанный в XML-описателе парсера в поле `profile_name`.
2. Переименуйте скопированный файл скрипта в файл `<имя_профиля>.js`.
3. Отредактируйте JS-скрипт парсера в соответствии с форматом кассовых чеков интегрированных терминалов.

При написании JS-парсеров действуют следующие соглашения:

1. JS-скрипт вызывается каждый раз при считывании новой строки чека. Скрипт производит синтаксический анализ строки и извлекает данные для отображения в титрах Модуля и передачи параметрам событий Модуля.
2. В назначенном для разбора чека JS-скрипте помещается функция-фабрика `createComponents`, создающая объект парсера. Метод `dispatcher` объекта парсера вызывается при поступлении каждой новой строки чека, передаваемой ему в качестве параметра (см. пример ниже).

**Листинг 2.** Пример скрипта парсера. В данном случае результатом скрипта является сдвиг на одну линию, без печати информации о чеке.

```
function createComponents() {
    var self = new Object();
    self.dispatcher = function(data) {
        listener.pushFilteredString(data); //скрипт разбора строки data
    }
    return self;
}
```

3. В JS-скрипте применяется специальный объект `listener`. Методы объекта `listener` используются для синтаксического разбора чека (см. **Методы объекта listener**) и уведомления объекта *POS Терминал* о распознанных чеках/продажах/событиях для последующего отображения в титрах.

### 3.2.1 Методы объекта listener

В данном разделе описаны методы объекта `listener`, доступные из скрипта парсера.

### 3.2.1.1 pushReceiptBegin

**Синтаксис:** pushReceiptBegin (code)

**Описание:** Обозначает начало нового чека

**Параметры:**

code	Код текущего чека из протокола, или null, если код чека недоступен из протокола. Тип аргумента: строка
------	--

**Возвращаемое значение:**

Идентификатор нового чека
---------------------------

### 3.2.1.2 pushCashierName

**Синтаксис:** pushCashierName (cashierName)

**Описание:** Устанавливает имя текущего кассира

**Параметры:**

cashierName	Имя текущего кассира. Тип аргумента: строка
-------------	---

### 3.2.1.3 pushNewSale

**Синтаксис:** pushNewSale (productName)

**Описание:** Добавляет новую запись о продаже товара в текущий чек

**Параметры:**

productName	Название товара. Тип аргумента: строка
-------------	--

### 3.2.1.4 pushSaleAmount

**Синтаксис:** pushSaleAmount (amount)

**Описание:** Устанавливает стоимость товара

**Параметры:**

amount	Величина стоимости товара. Тип аргумента: число
--------	---

### 3.2.1.5 pushSaleDiscount

**Синтаксис:** pushSaleDiscount (percents)

**Описание:** Устанавливает скидку на товар (в процентах)

**Параметры:**

percents	Величина скидки на товар (в процентах). Тип аргумента: число
----------	--

### 3.2.1.6 pushSaleProductsCount

**Синтаксис:** pushSaleProductsCount (count)

**Описание:** Устанавливает количество (вес) товара

**Параметры:**

count	Количество (вес) товара. Тип аргумента: число
-------	---

### 3.2.1.7 pushSaleArticle

**Синтаксис:** pushSaleArticle (article)

**Описание:** Устанавливает артикул товара

**Параметры:**

article	Артикул товара. Тип аргумента: строка
---------	---------------------------------------

### 3.2.1.8 pushSaleEnd

**Синтаксис:** pushSaleEnd ()

**Описание:** Обозначает окончание текущей продажи

**Параметры:** нет

### 3.2.1.9 pushReceiptDiscount

**Синтаксис:** pushReceiptDiscount (percents)

**Описание:** Устанавливает общую скидку на текущую продажу (в процентах).

**Внимание!** Данная операция отменяет скидки на отдельные товары.

**Параметры:**

percents	Величина общей скидки на сумму чека (в процентах). Тип аргумента: число
----------	---

### 3.2.1.10 pushReceiptEnd

**Синтаксис:** pushReceiptEnd (totalAmount)

**Описание:** Обозначает окончание текущей продажи

**Параметры:**

totalAmount	Итоговая сумма чека из протокола, или null. В случае значения null итоговая сумма чека устанавливается, исходя из суммы всех продаж внутри чека (так же, как и в методе pushReceiptEnd()). Тип аргумента: число
-------------	---

**Синтаксис:** pushReceiptEnd ()

**Описание:** Обозначает окончание текущей продажи. Итоговая сумма чека устанавливается исходя из суммы всех продаж внутри чека

**Параметры:** нет

### 3.2.1.11 pushFilteredString

**Синтаксис:** pushFilteredString (line)

**Описание:** Добавляет к титрам строку. После этого происходит переход на новую строку в титрах

**Параметры:**

line	Символ строки чека в титрах. Тип аргумента: строка
------	--

### 3.2.1.12 pushEvent

**Синтаксис:** pushEvent (eventDescription)

**Описание:** Посылает события в систему SecurOS от имени объекта *POS Терминал Модуля* (список событий см. в [Руководство пользователя SecurOS POS](#))

**Параметры:**

eventDescription	Название события. Тип аргумента: строка
------------------	---

**Возвращаемое значение:**

	Идентификатор события
--	-----------------------

**Синтаксис:** pushEvent (eventDescription, cashierName, amount)

**Описание:** Посылает события в систему SecurOS от имени объекта *POS Терминал Модуля* (список событий см. в [Руководство пользователя SecurOS POS](#))

**Параметры:**

eventDescription	Название события. Тип аргумента: строка. receiptCode — номер чека для данного события, или null, если номер чека не определен для данного события. Тип аргумента: строка
------------------	--

cashierName	Имя кассира для данного события или null, если имя кассира не определено для данного события. Тип аргумента: строка
-------------	---

amount	Сумма чека (продажи) для данного события, или null, если сумма чека (продажи) неопределена для данного события. Тип аргумента: число
<b>Возвращаемое значение:</b>	
	Идентификатор события

### 3.2.1.13 setEventReceiptCode

**Синтаксис:** setEventReceiptCode (eventId, receiptCode)

**Описание:** Устанавливает код чека для события с данным eventId

**Параметры:**

eventId	Идентификатор события. Тип аргумента: число
receiptCode	Код чека. Тип аргумента: строка.

### 3.2.1.14 setReceiptCode

**Синтаксис:** setReceiptCode (receiptId, receiptCode, updateSalesCode)

**Описание:** Устанавливает код чека с данным receiptId. Если updateSalesCode == true, то устанавливает код чека и для всех продаж чека, иначе устанавливает код только для самого чека

**Параметры:**

receiptId	Идентификатор чека. Тип аргумента: число
receiptCode	Код чека. Тип аргумента: строка
updateSalesCode	Логическое условие (см. описание метода). Тип аргумента: логический

## 4 Пример интеграции

Ниже представлен пример JS-скрипта интеграции кассового терминала.

### Листинг 3. Исходный чек

```
* ООО "Авалон" *
=====
Кассир: Иванова Ольга
=====
+ Рис 1 кг 100.0
+ Пельмени 1 кг 50.0
=====
СКИДКА: 10 \% 15.0
ИТОГО: 135.0
* Ждем Вас снова! *
```

Последовательность действий:

1. Создайте файл парсера <имя\_профиля>.js и добавьте нижеприведенные фрагменты кода.
2. Определение начала и конца чека.

### Листинг 4. Определение начала чека

```
self.testReceiptBegin = function(data) {
  if (/.* ООО "Авалон" \*/.test(data)) {
    listener.pushReceiptBegin();
    listener.pushFilteredString("-----");
    listener.pushFilteredString(" Начало чека ");
    listener.pushFilteredString("-----");
    return true;
  }
  return false;
}
```

На второй строке вышеприведенного кода производится проверка соответствия (метод `test`) пришедшей строки и строки, определяющей начало чека. В строках с 3 по 7 описываются действия, производимые в случае найденного начала чека: отправка сообщения объекту *POS Терминал Модуль* о найденном начале чека (строка 3), и наложение титров на привязанные камеры (строки с 4 по 6). Аналогичным образом производится определение конца чека.

### Листинг 5. Определение конца чека

```
self.testReceiptEnd = function(data) {
  if (/.* Ждем Вас снова! \*/.test(data)) {
    listener.pushReceiptEnd();
    listener.pushFilteredString("-----");
    listener.pushFilteredString(" Конец чека ");
    listener.pushFilteredString("-----");
    return true;
  }
}
```

```
    return false;
}
```

3. Добавьте код определения кассира, пробивающего чек.

#### Листинг 6. Определение кассира

```
self.testCashierName = function(data) {
    var matches = /Кассир: (\w+\s*\w*)/.exec(data);
    if (matches != null && matches.length >= 2) {
        listener.pushCashierName(matches[1]);
        return true;
    }
    return false;
}
```

Выделение фамилии и имени кассира происходит с помощью стандартного синтаксиса регулярных выражений (см. строку 2 вышеприведенного кода, метод `exec`). В случае, если пришедшая строка совпадает с написанным шаблоном, производится регистрация текущего активного кассира (строка 4). Для получения подробной справочной информации о регулярных выражениях Javascript см. [http://developer.mozilla.org/en/docs/Core\\_JavaScript\\_1.5\\_Reference](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference).

4. Добавьте код определения отдельных продаж.

#### Листинг 7. Определение продаж

```
self.testSale = function(data) {
    var matches = /\+\s(\w+)\s+(\d+)\s*\w*\s*(\d+\.\?\d*)/.exec(data);
    if (matches != null && matches.length >= 3) {
        var productName = matches[1];
        var productCount = parseFloat(matches[2]);
        var amount = parseFloat(matches[3]);
        listener.pushNewSale(productName);
        listener.pushSaleProductsCount(productCount);
        listener.pushSaleAmount(amount);
        listener.pushSaleEnd();
        listener.pushFilteredString(data);
        return true;
    }
    return false;
}
```

Выделение параметров продажи происходит с помощью стандартного синтаксиса регулярных выражений (см. строку 2 вышеприведенного кода). В случае, если пришедшая строка совпадает с написанным шаблоном, производится регистрация имени продукта (строка 7), количества товара (строка 8), суммы продажи (строка 9), а также посылается сигнал о завершении продажи товара.

5. Добавьте код определения количества (или веса) текущего товара.

#### Листинг 8. Определение стоимости всех товаров

```
self.testTotalSale = function(data) {
```

```

var matches = /ИТОГО: \s+(\w*)/.exec(data);
if (matches != null) {
var totalCount = parseFloat(matches[1]);
listener.pushSaleProductsCount(totalCount);
return true;
}
return false;
}

```

Выделение итоговой суммы происходит с помощью стандартного синтаксиса регулярных выражений (см. строку 2 вышеприведенного кода). В случае, если пришедшая строка совпадает с написанным шаблоном, производится регистрация общей суммы по чеку (строка 5). Аналогичным образом производится определение общей скидки по чеку (в процентах).

#### Листинг 9. Определение скидки по чеку

```

self.testDiscount = function(data) {
var matches = /СКИДКА: \s+(\w+)\s+\%?\s+(\d+\.\d*)/.exec(data);
if (matches != null && matches.length >= 3) {
var saleDiscount = parseFloat(matches[1]);
listener.pushReceiptDiscount(saleDiscount);
return true;
}
return false;
}

```

#### 6. Добавляем указанные функции в основной скрипт:

**Листинг 10.** Основной текст скрипта. Обработка строки заключается в сдвиге строки, печатании символа строки в титре и применении всех созданных функций к поступившей строке чека для извлечения данных.

```

function createComponents() {
var self = new Object();
... //функции разбора чека
self.dispatcher = function(data) { //метод dispatcher обработки
out.println(data); //поступающей строки
listener.pushFilteredString(data);
this.testReceiptBegin(data) ||
this.testReceiptEnd(data) ||
this.testCashierName(data) ||
this.testSale(data) ||
this.testTotalSale(data) ||
this.testDiscount(data);
}
return self;
}

```

Итоговый листинг скрипта интеграции представлен ниже.

#### Листинг 11. Итоговый листинг

```

function createComponents() {
var self = new Object();
self.testReceiptBegin = function(data) {

```



```
if (/\/* 000 "Авалон" \*/.test(data)) {
    listener.pushReceiptBegin();
    listener.pushFilteredString("-----");
    listener.pushFilteredString(" Начало чека ");
    listener.pushFilteredString("-----");
    return true;
}
return false;
}

self.testReceiptEnd = function(data) {
    if (/\/* Ждем вас снова! \*/.test(data)) {
        listener.pushReceiptEnd();
        listener.pushFilteredString("-----");
        listener.pushFilteredString(" Конец чека ");
        listener.pushFilteredString("-----");
        return true;
    }
    return false;
}

self.testSale = function(data) {
    var matches = /\+\/s(\w+)\s+(\d+)\s*\w*\s*(\d+\.\?\d*)/.exec(data);
    if (matches != null && matches.length >= 3) {
        var productName = matches[1];
        var productCount = parseFloat(matches[2]);
        var amount = parseFloat(matches[3]);
        listener.pushNewSale(productName);
        listener.pushSaleProductsCount(productCount);
        listener.pushSaleAmount(amount);
        listener.pushSaleEnd();
        listener.pushFilteredString(data);
        return true;
    }
    return false;
}

self.testTotalSale = function(data) {
    var matches = /\ИТОГО: \s+(\w*)/.exec(data);
    if (matches != null) {
        var totalCount = parseFloat(matches[1]);
        listener.pushSaleProductsCount(totalCount);
        return true;
    }
    return false;
}

self.testDiscount = function(data) {
    var matches = /\СКИДКА: \s+(\w+)\s+\%?\s+(\d+\.\?\d*)/.exec(data);
    if (matches != null && matches.length >= 3) {
        var saleDiscount = parseFloat(matches[1]);
        listener.pushReceiptDiscount(saleDiscount);
        return true;
    }
    return false;
}

self.testCashierName = function(data) {
    var matches = /\Кассир: (\w+\s*\w*)/.exec(data);
    if (matches != null && matches.length >= 2) {
```

```
        listener.pushCashierName(matches[1]);
        return true;
    }
    return false;
}

self.dispatcher = function(data) {
    out.println(data);
    listener.pushFilteredString(data);
    this.testReceiptBegin(data) ||
    this.testReceiptEnd(data) ||
    this.testCashierName(data) ||
    this.testSale(data) ||
    this.testTotalSale(data) ||
    this.testDiscount(data);
}
return self;
}
```

## 5 Приложения

### 5.1 Приложение 1. Информация для Службы технической поддержки

Данный раздел описывает требования к служебной информации, необходимой при обращении в Службу технической поддержки компании Intelligent Security Systems.

---

**Примечание.** Собранные сведения необходимо направлять в Службу технической поддержки (см. раздел [Обращение за технической поддержкой](#)).

---

Для более скорого разрешения проблем подготовьте следующую техническую информацию:

**Внимание!** Сведения в пунктах, отмеченных знаком "\*", являются обязательными для предоставления.

1. (\*) Ф. И. О.
2. (\*) Название организации.
3. (\*) Контактная информация: телефон, e-mail.
4. Если Вы являетесь партнером Intelligent Security Systems, то укажите, с каким менеджером компании Intelligent Security Systems Вы работаете; в ином случае, укажите следующие сведения:
  - Компания, в которой приобретался комплект программного и аппаратного обеспечения.
  - Действия для устранения проблемы, предложенные при обращении к партнеру, у которого приобретался комплект.
5. (\*) Описание проблемы (неполадки).
6. (\*) Описание действий, которые приводят к возникновению проблемы.
7. Описание изменений в настройках/конфигурации системы, которые привели к возникновению проблемы.
8. Системная и диагностическая информация о компьютере и конфигурации системы SecurOS, полученная с помощью утилиты **SystemInfo** (см. [Руководство администратора SecurOS](#) для подробной информации об использовании утилиты).

Если невозможно запустить данную утилиту, предоставьте следующую информацию:

- (\*) идентификаторы и даллас-коды используемых ключей Guardant;

---

**Примечание.** Даллас-коды оборудования можно просмотреть с помощью утилиты **ISS Hardware Report** (см. [Руководство администратора SecurOS](#) для подробной информации об использовании утилиты).

---

- (\*) наименование и версия установленного ПО производства компании Intelligent Security

Systems;

- общее количество видеосерверов и удаленных рабочих мест оператора в системе;
- операционная система (наименование платформы, версия сервисного пакета).

9. По возможности предоставьте любую другую полезную информацию, например:

- конфигурация компьютерного оборудования;
- загрузка центральных процессоров;
- объемы используемой оперативной и виртуальной памяти;
- загрузка сети;
- конфигурация сети и сетевого окружения.

# Предметный указатель

## Р

pushCashierName, метод, 10  
pushEvent, метод,  
    eventDescription, 12  
    eventDescription, cashierName, amount, 12  
pushFilteredString, метод, 12  
pushNewSale, метод, 10  
pushReceiptBegin, метод, 10  
pushReceiptDiscount, метод, 11  
pushReceiptEnd, метод,  
    totalAmount, 11  
    без параметров, 12  
pushSaleAmount, метод, 10  
pushSaleArticle, метод, 11  
pushSaleDiscount, метод, 10  
pushSaleEnd, метод, 11  
pushSaleProductsCount, метод, 11

## S

setEventReceiptCode, метод, 13  
setReceiptCode, метод, 13

## T

техническая поддержка,  
    обращение, 4  
    подготовка служебной информации, 19