



INTELLIGENT SECURITY SYSTEMS



POS-ИНСПЕКТОР

Руководство по интеграции оборудования POS Терминал

Версия 4.3.2

Руководство по интеграции оборудования POS Терминал (INTEG-Ru, сборка 3 от 26 февраля 2009 г.).

© Copyright ISS Technology 2005–2009

Отпечатано в России.

ISS Technology 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 ISS Technology предоставлять точную и актуальную информацию пользователям продукта.

Оглавление

Оглавление	3	2 Процедура интеграции	7
Предисловие	4	2.1 Создание XML-файла описателя парсера	7
Назначение	4	2.2 Реализация JS-парсера	8
Целевая аудитория	4	2.2.1 Методы listener	9
Использование руководства	4	3 Пример интеграции	12
Обращение за технической поддержкой .	4	Предметный указатель	18
1 Общие сведения	5		

Предисловие

Назначение

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

Целевая аудитория

Руководство предназначено для системных инсталляторов, интеграторов и администраторов — опытных пользователей операционной системы Microsoft® Windows®, имеющих навыки по установке аппаратного обеспечения и построению сети на основе протокола TCP/IP и знающих основы конфигурации системы безопасности SecurOS.

Использование руководства

Данный документ организован так, что его можно использовать как в печатном, так и в электронном виде. В последнем случае можно воспользоваться такими возможностями ПО Adobe Reader как закладки и гипертекстовые ссылки для навигации по документу. Данное руководство ссылается на другие документы по SecurOS ([Руководство администратора SecurOS](#), [Руководство по установке SecurOS](#)). Эти документы можно найти на установочном диске SecurOS либо на веб-сайте компании ISS (www.iss.ru).

Обращение за технической поддержкой

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

За дальнейшей информацией обращайтесь в Службу технической поддержки компании ISS:

- по телефону:
+7 (495) 734 33 33 (многоканальный, с понедельника по четверг с 10:00 до 19:00, в пятницу с 10:00 до 18:00 по московскому времени);
- по электронной почте:
support@iss.ru.

1. Общие сведения

Интеграция кассовых терминалов с подсистемой POS-Инспектор 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:

1. Установите ПО системы безопасности SecurOS (см. [Руководство по установке SecurOS](#)).
2. Установите ПО модуля POS-Инспектор. Создайте и настройте необходимые объекты модуля и объекты системы SecurOS. По вопросам конфигурации системы и установки (подключения) кассовых терминалов рекомендуем обратиться к [Руководству пользователя POS-Инспектор](#).
3. Создайте шаблоны для составления скрипта синтаксического анализатора (парсера) чеков (подробное описание процедуры см. в [2 Процедура интеграции](#) на стр. 7).
4. Запустите SecurOS.
5. Настройте объект *POS Терминал*. В окне настройки объекта в поле **Профиль разборщика** выберите имя созданного файла парсера (рис. 1).

Настройки подключения

☒ Сетевое

TCP Порт: 1

☐ RS232

Порт:

Скорость (бит/с):

Биты данных:

Четность:

Стоповые биты:

Управление потоком:

☐ DTR ☐ RTS

Профиль разборщика:

Frontol

База данных:

База данных 1

Рис. 1. Окно свойств объекта POS Терминал

2. 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 (см. [2.1 Создание XML-файла описателя парсера](#) на стр. 7).
2. Реализация JS-парсера (см. [2.2 Реализация JS-парсера](#) на стр. 8).

2.1 Создание XML-файла описателя парсера

В подкаталоге \POSParsers основной 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:

1. Скопируйте один из XML-описателей парсера в файл с названием, соответствующим новому парсеру.
2. Модифицируйте файл XML-описателя нового парсера:
 - a) В поле атрибута **profile_name** пропишите путь к JS-файлу, относительный к корневому каталогу SecurOS.
 - b) В поле атрибута **charsetname** укажите тип кодировки поступаемых данных 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!

2.2 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.

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

Чтобы создать JS-скрипт парсера:

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.

Процедура интеграции

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. [2.2.1 Методы listener](#) на стр. 9) и уведомления объекта *POS Терминал* о распознанных чеках/продажах/событиях для последующего отображения в титрах.

2.2.1 Методы listener

Таблица 1. Методы объекта `listener`, доступные из скрипта парсера

Метод	Параметры	Описание
<code>pushReceiptBegin()</code>	Отсутствуют.	Обозначает начало нового чека. <i>Внимание!</i> Метод устарел. Используйте метод <code>pushReceiptBegin(code)</code> вместо данного метода.
<code>pushReceiptBegin (code)</code>	<code>code</code> — код текущего чека из протокола, или <code>null</code> , если код чека недоступен из протокола. Тип аргумента: строка	Обозначает начало нового чека. Возвращает идентификатор нового чека.
<code>pushCashierName (cashierName)</code>	<code>cashierName</code> — имя текущего кассира. Тип аргумента: строка	Устанавливает имя текущего кассира.
<code>pushNewSale (productName)</code>	<code>productName</code> — название товара. Тип аргумента: строка	Добавляет новую запись о продаже товара в текущий чек.

(продолжение на следующей странице)

Процедура интеграции

(начало на предыдущей странице)

Метод	Параметры	Описание
<code>pushSaleAmount (amount)</code>	<code>amount</code> — величина стоимости товара. Тип аргумента: число	Устанавливает стоимость товара.
<code>pushSaleDiscount (percents)</code>	<code>percents</code> — величина скидки на товар (в процентах). Тип аргумента: число	Устанавливает скидку на товар (в процентах).
<code>pushSaleProductsCount (count)</code>	<code>count</code> — количество (вес) товара. Тип аргумента: число	Устанавливает количество (вес) товара.
<code>pushSaleArticle (article)</code>	<code>article</code> — величина артикула товара. Тип аргумента: строка	Устанавливает артикул товара.
<code>pushSaleEnd ()</code>	Отсутствуют.	Обозначает окончание текущей продажи.
<code>pushReceiptDiscount (percents)</code>	<code>percents</code> — величина общей скидки на сумму чека (в процентах). Тип аргумента: число	Устанавливает общую скидку на текущую продажу (в процентах). <i>Внимание!</i> Эта операция отменит скидки на отдельные товары.
<code>pushReceiptEnd (totalAmount)</code>	<code>totalAmount</code> — итоговая сумма чека из протокола, или <code>null</code> . В случае значения <code>null</code> итоговая сумма чека устанавливается, исходя из суммы всех продаж внутри чека (так же, как и в методе <code>pushReceiptEnd()</code>). Тип аргумента: число	Обозначает окончание текущей продажи.
<code>pushReceiptEnd ()</code>	Отсутствуют.	Обозначает окончание текущей продажи. Итоговая сумма чека устанавливается исходя из суммы всех продаж внутри чека.
<code>pushFilteredString (line)</code>	<code>line</code> — символ строки чека в титрах. Тип аргумента: строка	Добавляет к титрам строку. После этого происходит переход на новую строку в титрах.

(продолжение на следующей странице)

Процедура интеграции

(начало на предыдущей странице)

Метод	Параметры	Описание
<code>pushEvent (eventDescription)</code>	<code>eventDescription</code> — название события. Тип аргумента: строка	Посылает события в систему SecurOS от имени объекта <i>POS Терминал Модуля</i> (список событий см. в Руководстве пользователя POS-Инспектор). Возвращает идентификатор события.
<code>pushEvent (eventDescription, cashierName, amount)</code>	<code>eventDescription</code> — название события. Тип аргумента: строка. <code>receiptCode</code> — номер чека для данного события, или <code>null</code> , если номер чека не определен для данного события. Тип аргумента: строка. <code>cashierName</code> — имя кассира для данного события или <code>null</code> , если имя кассира не определено для данного события. Тип аргумента: строка. <code>amount</code> — сумма чека (продажи) для данного события, или <code>null</code> , если сумма чека (продажи) неопределена для данного события. Тип аргумента: число	Посылает события в систему SecurOS от имени объекта <i>POS Терминал Модуля</i> (список событий см. в Руководстве пользователя POS-Инспектор). Возвращает идентификатор события.
<code>setEventReceiptCode (eventId, receiptCode)</code>	<code>eventId</code> — идентификатор события. Тип аргумента: число. <code>receiptCode</code> — код чека. Тип аргумента: строка.	Устанавливает код чека для события с данным <code>eventId</code>
<code>setReceiptCode (receiptId, receiptCode, updateSalesCode)</code>	<code>receiptId</code> — идентификатор чека. Тип аргумента: число. <code>receiptCode</code> — код чека. Тип аргумента: строка. <code>updateSalesCode</code> — логическое условие (см. описание метода). Тип аргумента: логический	Устанавливает код чека с данным <code>receiptId</code> . Если <code>updateSalesCode == true</code> , то устанавливает код чека и для всех продаж чека, иначе устанавливает код только для самого чека.

3. 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 Модуля* 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.

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 *(начало)*

```
function createComponents() {
    var self = new Object();

    self.testReceiptBegin = function(data) {
        if (/\/\* ООО "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;
    }
}
```

Листинг 11. Итоговый листинг *(окончание)*

```
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;
}
```

Предметный указатель

A

Adobe Reader, [4](#)

P

POS Терминал, объект, [5](#), [9](#), [11](#), [12](#)

pushCashierName, функция, [9](#)

pushEvent, функция, [11](#)

pushFilteredString, функция, [10](#)

pushNewSale, функция, [9](#)

pushReceiptBegin, функция, [9](#)

pushReceiptDiscount, функция, [10](#)

pushReceiptEnd, функция, [10](#)

pushSaleAmount, функция, [10](#)

pushSaleArticle, функция, [10](#)

pushSaleDiscount, функция, [10](#)

pushSaleEnd, функция, [10](#)

pushSaleProductsCount, функция, [10](#)

S

SecurOS, [4](#), [5](#)

setEventReceiptCode, функция, [11](#)

setReceiptCode, функция, [11](#)

T

TCP/IP-протокол, [4](#)

и

интеграция

возможности оператора, [5](#)

последовательность действий, [5](#)

к

конфигурации, [5](#)

п

парсер, [7](#), [8](#)

DTD-документ, [7](#)

profile_name, [8](#)

XML-файл, [7](#)

модификации, [7](#)

объект listener, [9](#)

методы listener, [9](#)

описатели, [7](#)

реализация, [8](#)

создание описателя, [8](#)

подключение кассовых терминалов, [5](#)

б

реализация парсера, [7](#)

в

создание парсера, [7](#)

г

техническая поддержка, [4](#)

