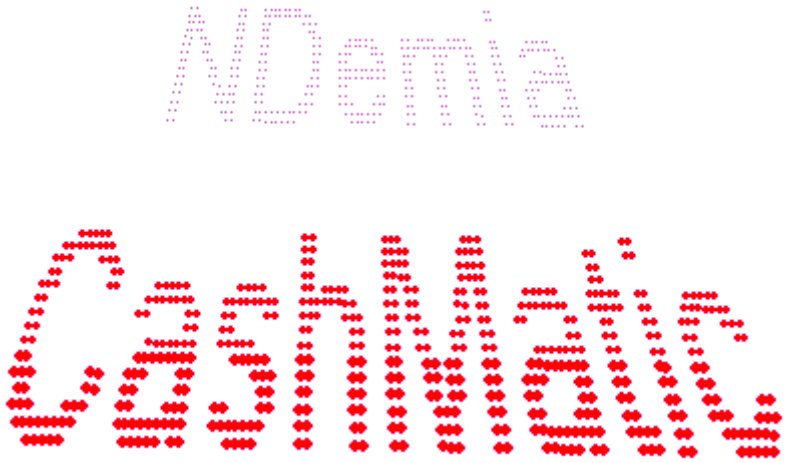

NDemia CashMatic KioskBrowser. Руководство прикладного программиста.

Содержание

1 Общий обзор

Программа **NDemia CashMatic KioskBrowser** разработана для организации работы терминалов приёма платежей.
NDemia CashMatic KioskBrowser является расширенной оболочкой для **Internet Explorer** и, соответственно, включает в себя всю функциональность **Internet Explorer** как средства просмотра HTML-контента, включая поддержку встроенных в **Internet Explorer** скриптовых языков (javascript и Visual Basic Script). Никаких специальных ограничений на контент не накладывается, поэтому вполне нормально поддерживается и работа скриптов серверной стороны (PHP, ASP и другие подобные технологии).
С другой стороны, **NDemia CashMatic KioskBrowser** поддерживает работу со специализированным оборудованием, характерным для платёжных терминалов и не характерным для обычных компьютеров - купюроприёмник, чековый принтер, фискальный регистратор.
Кроме того, **NDemia CashMatic KioskBrowser** организует порядок просмотра контента более жёстко, чем обычные веб-браузеры, ограничивая возможные действия пользователя по управлению навигацией, а также выполняя переходы на фиксированные адреса при определённых сигналах от специального оборудования. Вместе с тем разработчику контента предоставляются дополнительные возможности по организации сеанса работы пользователя на клиентской стороне, что уменьшает зависимость работы пользовательского интерфейса от стабильности связи с сервером и сокращает дополнительный трафик, несущественный с точки зрения конечного результата. Управление сценарием приёма платежа на клиентской стороне является положительным моментом и с точки зрения информационной безопасности - уменьшается количество закрытых данных, которые требуется передавать по каналам связи.
И наконец, **NDemia CashMatic KioskBrowser** предоставляет возможность подключения разработанных сторонними программистами COM-объектов, реализующих соответствующие интерфейсы расширения.

2 Объектная модель

Центральным элементом объектной модели **NDemia CashMatic KioskBrowser** является объект `CashMatic`, через него программисту предоставляется доступ ко всем остальным объектам, поддерживаемым приложением.
Важная особенность: объект `CashMatic` существует в контексте работы всей программы в целом, а не отдельной html-страницы или скрипта. Таким образом, управляя его свойствами, программист может передавать данные со страницы на страницу, минуя такие механизмы как cookie, PHP-сеансы и т.п.
Доступ к объекту `CashMatic` программист получает через javascript-объект **`window.external`** (можно просто `external`). Объект `external` определён в **Internet Explorer** как предоставление доступа к дополнительной объектной модели, обеспечиваемой хост-приложением - в данном случае так и есть.
Программисту рекомендуется каждый скрипт начинать с отдельной глобальной декларации
```
var CashMatic=external;
```

Объект `CashMatic` через свои свойства предоставляет доступ к другим объектам, поддерживаемым программой:

- `CashMaticHost` - обеспечивает взаимодействие с хост-приложением для скриптов и компонентов расширения;
- `CashMaticSession` - управление сеансом пользователя (создание и уничтожение набора переменных, связанных с сеансом, оповещение компонентов расширения о начале, завершении и внутренних событиях сеанса);
- `CashMaticTerminal` - контекст работы терминала в целом;
- `CashMaticSound` - звуковая поддержка, обеспечиваемая **NDemia CashMatic KioskBrowser**;
- `CashMaticCash` - управление купюроприёмником, создание и управление сеансом приёма купюр;
- `CashMaticPrint` - управление чековым принтером или фискальным регистратором;

Кроме перечисленного, объектная модель **NDemia CashMatic KioskBrowser** включает в себя определения COM-интерфейсов, необходимых для подключения компонентов расширения:

- `ICashMaticPlugin` интерфейс подключения расширений (реализуется клиентским расширением)
- `ICashMaticSession` интерфейс объекта `CashMaticSession` (пользовательский сеанс, реализуется хост-приложением)
- `ICashMaticTerminal` интерфейс объекта `CashMaticTerminal` (контекст терминала, реализуется хост-приложением)
- `ICashMaticSessionExtender` интерфейс расширения для пользовательского сеанса (реализуется клиентским расширением)
- `ICashMaticSessionExtender2` интерфейс расширения для пользовательского сеанса (реализуется клиентским расширением)
- `ICashMaticTerminalExtender` интерфейс расширения для контекста терминала (реализуется клиентским расширением)
- `ICashMaticTerminalExtender2` интерфейс расширения для контекста терминала (реализуется клиентским расширением)

2.1 Объект CashMatic (window.external), свойства

Имя	Тип результата	Тип обращения	Назначение
Cash	объект **CashMaticCash**	*Свойство, только чтение*	взаимодействие с купюроприёмником
Host	объект **CashMaticHost**	*Свойство, только чтение*	взаимодействие с хост-приложением
Print	объект **CashMaticPrint**	*Свойство, только чтение*	взаимодействие с чековым принтером или фискальным регистратором
Session	объект **CashMaticSession**	*Свойство, только чтение*	сеанс работы пользователя
Sound	объект **CashMaticSound**	*Свойство, только чтение*	звуковая поддержка
Terminal	объект **CashMaticTerminal**	*Свойство, только чтение*	общий контекст терминала

2.2 Объект CashMaticHost (CashMatic.Host), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
AddErrorName(ErrorName)	Пустой	*Метод*	Добавляет **ErrorName** в список имён отказов, вызывающих переход на страницу неисправности (см. Модель обработки отказов)
Application	объект **CashMatic**	*Свойство, только чтение*	Возвращает ссылку на хост-приложение
Build	Число	*Свойство, только чтение*	Номер сборки **NDemia CashMatic KioskBrowser**
ExecScript(Script, [Language])	Число	*Метод*	Выполнить скрипт, заданный строкой **Script**, написанный на языке **Language**. Как **Script** задаётся прямой текст скрипта, параметр **Language** должен иметь значение `"javascript"` (по умолчанию), `"JScript"` или `"VBScript"`. Метод предназначен для компонентов расширения, поскольку они работают вне контекста текущего HTML-документа и не могут вызвать выполнение скрипта другим способом. **Совместимость:** не использовать в версиях **NDemia CashMatic** ранее **2.6.0**
GetSpecialURL(SpecialName)	Строка	*Метод*	Преобразует **SpecialName** в адрес перехода для навигации (см. Специальные URL)
Name	Строка	*Свойство, только чтение*	Возвращает строку `"NDemia CashMatic KioskBrowser"`
Navigate(URL)	Число	*Метод*	Переход навигации на заданный адрес **URL**. Метод предназначен для компонентов расширения, поскольку они работают вне контекста текущего HTML-документа и не могут вызвать переход другим способом. Возвращает ноль.
Path	Строка	*Свойство, только чтение*	Возвращает путь установки **NDemia CashMatic KioskBrowser** на локальной машине (по умолчанию `"\Program Files\NDemia\CashMatic"`)
Valid	Логический	*Свойство, только чтение*	Возвращает истинное значение
Version	Строка	*Свойство, только чтение*	Возвращает версию продукта в читаемом виде. Версия 2.7.2 возвращается как `"2.72"`

3 Подключение прикладных скриптов

см. также:	Объектная модель	Подключение прикладных клиентских расширений	Пример платёжного интерфейса

NDemia CashMatic KioskBrowser является расширенной оболочкой для **Internet Explorer** и, соответственно, включает в себя всю функциональность **Internet Explorer** как средства просмотра HTML-контента, включая поддержку встроенных в **Internet Explorer** скриптовых языков (**`javascript`** и **`Visual Basic Script`**). Никаких специальных ограничений на контент не накладывается, поэтому вполне нормально поддерживается и работа скриптов серверной стороны (**`PHP`**, **`ASP`** и другие подобные технологии).

Прикладной программист может использовать любые скрипты, как специально разработанные для объектной модели **NDemia CashMatic**, так и общецелевые, работающие на любом веб-браузере. В любом случае прикладному программисту следует ознакомиться со следующими разделами документации:

- Как обеспечить автоматический запуск своего скрипта - см. Автоматический запуск скрипта;
- Как организовать обработку ошибок на уровне скрипта (событие **`window.onerror`**) дружественно по отношению к **NDemia CashMatic KioskBrowser** - см. Обработка событий window.onerror.

Прикладной скрипт может получить доступ к объектной модели **NDemia CashMatic** и использовать ряд свойств и методов, специально разработанных для решения типовых задач создания платёжного интерфейса - см. Объектная модель, см. Сеанс приёма платежа.

Прикладной программист может участвовать в обработке отказов терминала и управлять реакциями на отказы, см Модель обработки отказов.

Одной из центральных задач **NDemia CashMatic** является работа с купюроприёмником (приём наличных). Прикладной программист может разработать на основе **NDemia CashMatic** собственный уникальный и неповторимый интерфейс приёма наличных денег - см. Взаимодействие с купюроприёмником.

В части организации печати чеков **NDemia CashMatic** предлагает прикладному программисту большой набор различных возможностей, при этом в значительной степени унифицированно рассматривается достаточно широкий спектр оборудования - от простых (по интерфейсу подключения) принтеров, работающих через COM-порт без всяких драйверов, до фискальных регистраторов. См. Печать.

NDemia CashMatic KioskBrowser предоставляет прикладному программисту ряд возможностей по озвучиванию платёжного интерфейса - вплоть до комбинирования голосовых фраз из звукозаписей отдельных словосочетаний и слов, включая готовое решение для озвучивания чисел и денежных сумм. См. Звуковая поддержка.

Достаточно полно сформулированный пример платёжного интерфейса, который может работать на основе **NDemia CashMatic** - см. Пример платёжного интерфейса.

И наконец, если разработчику конечного прикладного продукта требуется использовать собственный исполнимый код в контексте приёма платежа - **NDemia CashMatic** имеет соответствующий механизм подключения сторонних DLL (компонентов расширения, или плагинов) - см. Подключение прикладных клиентских расширений, а также демонстрационный проект http://www.ndemia.com/CashMatic/plugins/Demo.

3.1 Стартовая страница

см. также:	Подключение прикладных скриптов	Модель навигации	Автоматический запуск скрипта
	Автоматические переходы	Специальные URL	Пример платёжного интерфейса

Стартовая страница - это HTML-документ, который автоматически открывается при запуске программы **NDemia CashMatic KioskBrowser**. Кроме того, на эту страницу делается автоматический переход со страницы неисправности после восстановления нормальной работы терминала (см. Модель обработки отказов).
Стартовая страница никак не связана с "домашней страницей" браузера (**Internet Explorer**) - см. Модель навигации.

Адрес стартовой страницы имеет свою постоянную настройку в системном реестре **Windows**, он доступен прикладным скриптам как специальный URL `"start"` - см. Специальные URL. Любой скрипт в **NDemia CashMatic KioskBrowser** может сослаться или перейти на стартовую страницу по её внутреннему адресу: **`"CashMatic:start"`** (см. Протокол "CashMatic:")

Прикладной скрипт, реализующий платёжный интерфейс, должен запускаться автоматически при включении терминала, поэтому разработчику скрипта требуется обеспечить настройку адреса стартовой страницы на свой скрипт - см. Автоматический запуск скрипта.

При отладке или технологических работах может потребоваться запуск **NDemia CashMatic KioskBrowser** с альтернативным стартом (изменённым начальным стартовым адресом). **NDemia CashMatic KioskBrowser** позволяет указать начальный стартовый адрес непосредственно при запуске, через параметр командной строки - см. Открытие HTML-страниц через параметр командной строки.
Обратите внимание: через параметр командной строки указывается только начальный стартовый адрес, т.е. самая первая открытая HTML-страница. Специальный URL `"start"` от этого не изменяется, альтернативный стартовый скрипт имеет возможность перейти на обычный стартовый адрес **`"CashMatic:start"`**.
Для удобства запуска **NDemia CashMatic KioskBrowser** с альтернативными стартовыми скриптами при установке программы регистрируется особый тип файлов (расширение имени файлов) - **CMHT** (CashMatic HTML). Подразумевается, что CMHT - это HTML со скриптами, разработанными специально для **NDemia CashMatic KioskBrowser**. Файлы с расширением CMHT открываются через запуск **NDemia CashMatic KioskBrowser**, при их открытии, например, в Проводнике **Windows** - в отличие от обычных HTML, запускающих веб-браузер по умолчанию - см. Расширение имён файлов CMHT.

3.1.1 Автоматический запуск скрипта

см. также:	Стартовая страница	Подключение прикладных скриптов	Автоматические переходы
	Обработка событий window.onerror	Пример платёжного интерфейса	Специальные URL
	Модель обработки отказов	Модель навигации	Протокол "CashMatic:"
	Платёжный сеанс	Приём купюр	Печать чеков

Для автоматического запуска скрипта при загрузке **NDemia CashMatic KioskBrowser** нужно назначить соответствующую HTML-страницу стартовой.
По умолчанию (при самой первой установке **NDemia CashMatic**, без дополнительных настроек реестра) стартовой страницей является **%ProgramFiles%\NDemia\CashMatic\HTML\start.html** (при самой первой установке в папке **%ProgramFiles%\NDemia\CashMatic\HTML** размещается Пример платёжного интерфейса).
Если разработчик разместит свою страницу в папке **%ProgramFiles%\NDemia\CashMatic\HTML** и даст ей имя **start.html**, то она будет автоматически открываться при каждом запуске **NDemia CashMatic KioskBrowser** (а также при восстановлении терминала из состояния ошибки).
Кроме этого, разработчик может создать страницу, на которую будет выполняться автоматический переход при ошибках терминала - **failure.html**.
Также можно создать страницу, на которую будет разрешён переход при снятии стекера купюроприёмника - **collect.html** (этот переход не делается автоматически, но это единственный адрес, на который разрешён переход при снятом стекере).

Дополнительные сведения о специальных адресах HTML-страниц: см. Специальные URL.

Рекомендуемый способ организации имён HTML-файлов для разработки платёжного интерфейса на основе **NDemia CashMatic**:

1. в папке **%ProgramFiles%\NDemia\CashMatic\HTML** прикладной программист должен создать собственную папку, например **MyFiles** (**%ProgramFiles%\NDemia\CashMatic\HTML\MyFiles**), и разместить в ней все свои файлы (**html**, **js**, **css**, и т.п.), возможно, в виде собственной структуры подпапок;
2. в системном реестре в разделе
HKEY_LOCAL_MACHINE\SOFTWARE\NDemia\CashMatic\URL
нужно создать строковый параметр с именем
urlroot
и значением:
%ProgramFiles%\NDemia\CashMatic\HTML\MyFiles
(теперь **NDemia CashMatic KioskBrowser** будет все файловые адреса определять внутри папки разработчика (**MyFiles**))

3.1.2 Открытие HTML-страниц через параметр командной строки

см. также: Стартовая страница | Тип файлов CMHT | Автоматический запуск скрипта | Специальные URL | Протокол "CashMatic:" | Подключение прикладных скриптов

При запуске программы **NDemia CashMatic KioskBrowser** можно указать альтернативный стартовый URL (имя HTML-файла стартовой страницы). Для этого имя файла должно быть указано как параметр командной строки, например:
`KioskBrowser f:\MyFiles\debug.html`
- таким способом можно запустить **KioskBrowser** с открытием файла **`f:\MyFiles\debug.html`**, который может содержать, например, специальную отладочную стартовую страницу.

Обратите внимание: через параметр командной строки указывается только начальный стартовый адрес, т.е. самая первая открытая HTML-страница. Специальный URL `"start"` от этого не изменяется, альтернативный стартовый скрипт имеет возможность перейти на обычный стартовый адрес **`"CashMatic:start"`**.

Совместимость: указание имени альтернативного стартового файла через командную строку поддерживается в **NDemia CashMatic**, начиная с версии **2.6.1**

Альтернативный стартовый файл может указываться как имя файла, как полное имя (с путём, как абсолютным, так и относительно текущей директории при запуске программы), как URL-ссылка, и т.п. Если файл не найден и указан без URL-протокола, то делается попытка разрешить ссылку через URL-протокол **`"CashMatic:"`** (см. Протокол "CashMatic:").

Совместимость: при указании имени альтернативного стартового файла через командную строку интерпретация в различных подразумеваемых форматах поддерживается, начиная с версии **2.7.0**.

3.1.3 Расширение имён файлов CMHT

см. также: Стартовая страница | Специальные URL | Протокол "CashMatic:" | Подключение прикладных скриптов | Автоматический запуск скрипта | Пример платёжного интерфейса

NDemia CashMatic регистрирует в **Windows** расширение имён файлов (тип файлов) **".cmht"** (CashMatic HTML). Файлы **`*.CMHT`** - это обычные HTML-файлы, которые на терминале ассоциированы с приложением **NDemia CashMatic KioskBrowser**, могут открываться простым щелчком мыши в Проводнике **Windows** (и другими аналогичными способами). Это сделано только для удобства - возможности открытия **KioskBrowser** в специальных режимах прямо шелчком по значку, других отличий для CMHT по сравнению с HTML не предусматривается.
В Пример платёжного интерфейса (папка **`%ProgramFiles%\NDemia\CashMatic\HTML`**) добавлен файл `start.cmht`, который является полной копией `start.html`, но через него щелчком можно запустить **NDemia CashMatic KioskBrowser**.

Совместимость: регистрация типа файлов CMHT поддерживается в **NDemia CashMatic**, начиная с версии **2.7.0**

3.2 Обработка событий window.onerror

см. также:	Подключение прикладных скриптов	Модель обработки отказов	Компонент расширения для CashMatic.Terminal

Обработчик ошибок **NDemia CashMatic KioskBrowser** автоматически устанавливается как свойство **window.onerror** (в терминах объектной модели **Internet Explorer**) при загрузке каждого HTML-документа.

Совместимость: Собственный обработчик ошибок **NDemia CashMatic KioskBrowser** поддерживается, начиная с версии **2.6.0**.

Прикладной скрипт, устанавливающий собственную обработку window.onerror, перехватывает внутреннюю обработку ошибок **NDemia CashMatic KioskBrowser**.
Для обеспечения нормальной работы приложения прикладной скрипт должен сохранить значение свойства window.onerror на момент запуска скрипта, и в случае вызова события window.onerror использовать сохранённое значение для вызова обработчика **NDemia CashMatic KioskBrowser**.

Пример:

```
var PrevOnError;
function Пример()
        //      эта функция устанавливает собственный обработчик ошибок
        //прикладного скрипта, сохраняя предыдущее значение как PrevOnError
{
        PrevOnError=onerror;    //сохраняем предыдущее значение
        onerror=ErrorHandler;   //устанавливаем наш обработчик (ErrorHandler)
}

function ErrorHandler(msg, url, line)
{
        alert("Ошибка: " + msg + " (\"" + url + "\": " + line + ")");
                //наша собственная обработка ошибки
                //(до всех остальных обработчиков)

        if(PrevOnError) //проверяем, был ли установлен предыдущий
                        //обработчик ошибок до нашего
                return PrevOnError(msg, url, line);
                        //вызываем предыдущий обработчик ошибок
        else
                //предыдущий обработчик не был установлен
                return true;    //возвращаем истинное значение для подавления
                        //диалога ошибки Internet Exporer
                        //(ложное значение разрешит обычную обработку
                        //ошибки Internet Explorer)
}
```

4 Подключение прикладных клиентских расширений

При запуске **NDemia CashMatic KioskBrowser** программа открывает раздел системного реестра **Windows**
HKEY_LOCAL_MACHINE\SOFTWARE\NDemia\CashMatic\Plugins
и перебирает все его подразделы.
В каждом подразделе ищется значение типа **REG_SZ** с именем
CLSID
Если такое значение задано, то оно воспринимается как идентификатор COM-класса компонента расширения.
Например:

```
[HKEY_LOCAL_MACHINE\Software\NDemia\CashMatic\Control Plugins\My Plugin]
"CLSID"="{06D4C323-931E-4279-9C9F-E493A2834328}"
```

CLSID - это единственное зарезервированное имя для подраздела расширения, разработчик расширения может создавать в этом подразделе другие значения и подразделы.

Если значение CLSID существует, то **NDemia CashMatic KioskBrowser**

1. создаёт объект класса CLSID
2. запрашивает у этого объекта интерфейс ICashMaticPlugin
3. вызывает метод ICashMaticPlugin::Init
4. удаляет объект

на этом подключение расширения с точки зрения хост-приложения заканчивается.
Компонент расширения через метод Init получает ссылку IDispatch-интерфейса объекта CashMatic, и далее через неё может установить свои расширители в CashMatic.Session и CashMatic.Terminal. Эти расширители будут получать оповещения о различных предопределённых событиях, а также о пользовательских событиях, инициируемых веб-контентом.

В синтаксисе C++ интерфейс ICashMaticPlugin определяется так:

```
class ICashMaticPlugin : public IDispatch
{
	public:
		virtual void CALLBACK Init(IDispatch* pCashMatic) = 0;
};

// {01288B75-B4EA-4d76-8322-8B5AC4E89995}
static const IID IID_ICashMaticPlugin = { 0x1288b75, 0xb4ea, 0x4d76,
	{ 0x83, 0x22, 0x8b, 0x5a, 0xc4, 0xe8, 0x99, 0x95 } };
```

Интерфейсы ICashMaticSession, ICashMaticSessionExtender, ICashMaticSessionExtender2 рассмотрены в описании объекта CashMaticSession.
Интерфейсы ICashMaticTerminal, ICashMaticTerminalExtender, ICashMaticTerminalExtender2 рассмотрены в описании объекта CashMaticTerminal.

5 Модель навигации

см. также:	Подключение прикладных скриптов	Стартовая страница	Страница неисправности
	Автоматические переходы	Специальные URL	Протокол "CashMatic:"
	Модель обработки отказов	Отложенные отказы	Контроль URL-адресов переходов

Порядок навигации (переходов) по HTML-страницам в **NDemia CashMatic KioskBrowser** несколько отличается от обычного для **Internet Explorer**:

- в некоторых особых состояниях терминала **NDemia CashMatic KioskBrowser** разрешает переходить только на определённые HTML-страницы, которые должны обрабатывать эти состояния (см. Контроль URL-адресов переходов);
- при некоторых событиях выполняются автоматические переходы на соответствующие HTML-страницы, предназначенные для обработки этих событий (см. Автоматические переходы).

Для указания адресов HTML-страниц, имеющих особое значение, используются специальные URL-адреса, настраиваемые в системном реестре **Windows** (см. Специальные URL).
Прикладные скрипты и компоненты расширения могут получать значения специальных URL-адресов, используя метод `GetSpecialURL` объекта `CashMaticHost`.
Кроме того, разработчик может ссылаться непосредственно на имена специальных URL, используя URL-протокол **`"CashMatic:"`** (см. Протокол "CashMatic:").

5.1 Контроль URL-адресов переходов

см. также:	Модель навигации	Автоматические переходы	Протокол "CashMatic:"
	Отложенные отказы	Компоненты расширения	Интерфейс ICashMaticTerminalExtender
	Специальные URL	Модель обработки отказов	Try Print Anyway

В некоторых особых состояниях терминала **NDemia CashMatic KioskBrowser** разрешает переходить только на определённые HTML-страницы, которые должны обрабатывать эти состояния.

Совместимость: адреса с URL-протоколом **`"javascript:"`** не считаются переходами, не подлежат контролю и не блокируются в особых состояниях терминала в **NDemia CashMatic**, начиная с версии **2.7.2**

В текущей версии такими состояниями являются инкассация и неисправность.
Состояние инкассации соответствует снятию/удалению стекера (кассы) купюроприёмника. В этом состоянии единственным разрешённым адресом перехода является специальный URL **`"collect"`** (**`"CashMatic:collect"`**).
Неисправность - наличие одного или нескольких отказов, требующих перехода на страницу неисправности (см. Модель обработки отказов). В этом состоянии единственным разрешённым адресом перехода является специальный URL **`"failure"`** (**`"CashMatic:failure"`**), за исключением случаев, когда в текущем сеансе приёма платежа уже приняты деньги — тогда при неисправности разрешены любые переходы (см. Отложенные отказы).

> ***Примечание:*** *В текущей версии инкассация имеет больший приоритет, чем неисправность - т.е. при возникновении неисправности в состоянии инкассации переход на страницу неисправности запрещён (но будет автоматически выполнен после восстановления купюроприёмника, если ошибка останется действующей). В будущих версиях это поведение может быть изменено.*

> ***Примечание:*** *В текущей версии при инкассации переход на* **`"CashMatic:collect"`** *не делается автоматически, но это единственный разрешённый адрес перехода при снятом стекере (переход может быть сделан компонентом расширения при вызове его метода* `PreCashCollect`*, см. интерфейс* `ICashMaticTerminalExtender`*).*

Прикладной программист может отключить контроль URL-адресов переходов:
раздел реестра
`HKEY_CURRENT_USER\Software\NDemia\CashMatic`
параметр **`"Try Start Anyway"`**
(тип **`REG_DWORD`**),
ненулевое значение отключает контроль переходов.
Рекомендуемое использование: компонент расширения (плагин), реализующий собственную расширенную обработку особых состояний (ошибка, инкассация), может временно отключить контроль переходов со стороны браузера, например, чтобы перейти на некоторую технологическую страницу, на которой обслуживающий персонал может получить подробную диагностику по ошибке и выполнить какие-либо тесты и корректирующие команды.
Не рекомендуется постоянное отключение контроля переходов, поскольку он встроен в программу с целью надёжного предотвращения возможности работы терминала в неисправном/неполнофункциональном состоянии (как своего рода предохранитель).

Совместимость: Параметр **`"Try Start Anyway"`** поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

5.2 Автоматические переходы

см. также:	Модель навигации	Специальные URL	Контроль URL-адресов переходов
	Стартовая страница	Модель обработки отказов	Отложенные отказы

При некоторых событиях **NDemia CashMatic KioskBrowser** выполняет автоматические переходы на соответствующие HTML-страницы, предназначенные для обработки этих событий. Автоматические переходы выполняются без действия пользователя (помимо или в обход действий пользователя)

В текущей версии выполняются автоматические переходы на стартовую страницу и страницу неисправности.
Стартовая страница открывается при запуске **NDemia CashMatic KioskBrowser**, сюда же делается переход после восстановления нормальной работы со страницы неисправности. Стартовая страница никак не связана с "домашней страницей" **Internet Explorer**. Стартовая страница определяется через специальный URL **`"start"`** (**`"CashMatic:start"`**).
В командной строке при запуске программы можно задать другой начальный стартовый адрес, см. Открытие HTML-страниц через параметр командной строки.

Страница неисправности открывается автоматически при возникновении отказа, являющегося ошибкой (за исключением случаев отложенного отказа, см. Модель обработки отказов). Рекомендуется всегда делать страницу неисправности локальной на терминале, чтобы отказ связи с сервером не ломал всё окончательно.
Страница неисправности определяется через специальный URL **`"failure"`** (**`"CashMatic:failure"`**).

5.3 Специальные URL

см. также:	Модель навигации	Автоматические переходы	Контроль URL-адресов переходов
	Стартовая страница	Модель обработки отказов	Протокол "CashMatic:"

NDemia CashMatic KioskBrowser использует специальные URL для контроля допустимости переходов и для выполнения автоматических переходов.
Специальные URL настраиваются в системном реестре **Windows**.

Используется раздел
HKEY_LOCAL_MACHINE\SOFTWARE\NDemia\CashMatic\URL
При необходимости получить значение специального URL выполняется следующая последовательность действий:

1. В указанном разделе реестра проверяется параметр, имя которого равно имени специального URL, тип **REG_SZ** (например, для специального URL start в указанном разделе реестра может быть задан строковый параметр **"start"**). Если такой параметр существует и имеет непустое значение, то далее используется его значение (например: **http://myserver/start.php**).
2. Если значение URL-адреса по п.1 не определилось, то рассматривается строковый параметр urlroot. Этот параметр задаёт общий корень (корневой путь) для всех специальных URL (по умолчанию **"%ProgramFiles%NDemia\CashMatic\HTML"**, но может быть и ссылка на интернет-ресурс). К корневому пути добавляется имя специального URL.

   Например, специальный URL start преобразуется в **"%ProgramFiles%\NDemia\CashMatic\HTML\start"**).

3. Если полученный по п.1 или п.2 URL-адрес не содержит префикса спецификации протокола (URL-схемы) (примеры префиксов спецификации протокола, или URL-схем: **"http:"**, **"CashMatic:"**, **"file:"**), то предполагается ссылка на локальный файл.
   В этом случае добавляется умолчательное расширение **".html"**, применяется подстановка значений вставленных в URL-адрес переменных окружения (*environment variables*, переменная name вставляется как **%name%**), например, можно в путь вставлять **%ProgramFiles%**, **%APPDATA%** и т.п.
   Если специальный URL был задан явным параметром (определился по п.1) как локальный файл с относительным путём, то полный путь определяется относительно локального корня HTML (по умолчанию **"%ProgramFiles%\NDemia\CashMatic\HTML"**).

   Пример 1: если параметр start имеет значение **"%ProgramFiles%/start"**, то специальный URL start преобразуется в **"C:\Program Files\start.html"**.
   Пример 2: если параметр start имеет значение **"my\start"**, то специальный URL start преобразуется в **"C:\Program Files\NDemia\CashMatic\html\my\start.html"**.

4. Выполняется преобразование URL-адреса, известное как *"каноникализация"* или *"нормализация"* (из URL-адреса корректно удаляются элементы **"."** и **".."**, выполняется раскодировка специальных знаков, недопустимых для интернет-адресов, и т.п.).

   Пример 1: если параметр start имеет значение **"..\my%20start"**, то специальный URL start преобразуется в **"C:\Program Files\NDemia\CashMatic\my start.html"**.
   Пример 2: если параметр start имеет значение **"http://example.com\CashMatic\.\.././start"**, то специальный URL start преобразуется в **"http://example.com/start"** (обратные слэши преобразованы в прямые, расширение имени файла не добавлено).
   Пример 3: если параметр start имеет значение **"example.com/CashMatic/./..\.\start"**, то специальный URL start преобразуется в **"C:\Program Files\NDemia\CashMatic\html\example.com\start.html"** (прямые слэши преобразованы в обратные, умолчательное расширение имени файла добавлено).

Совместимость: Предполагаемая по умолчанию ссылка на локальный файл, подстановка переменных окружения, каноникализация URL поддерживаются в **NDemia CashMatic**, начиная с версии **2.7.0**

Таким образом, прикладные скрипты и компоненты расширений могут через ссылки на очень простые по форме специальные URL ссылаться на достаточно сложные конструкции URL-адресов, указывающие как на локальные файлы, так и на интернет-ресурсы.
Приложения могут выполнять вышеописанное преобразование специальных URL через метод GetSpecialURL объекта CashMaticHost.
Используя URL-протокол "CashMatic:", приложения **NDemia CashMatic KioskBrowser** могут ссылать-

ся непосредственно на специальные URL по имени, вообще не занимаясь их преобразованием.
Прикладному программисту предоставляется механизм гибкой настройки путей и имён файлов на конкретном терминале без изменений ссылок в HTML и javascript-коде.
Использование системного реестра **Windows** для настройки специальных URL позволяет убрать из открытого кода ссылки на реальные имена и пути размещения файлов. Прикладной программист может добавить в реестр имена собственных специальных URL, которые могут быть использованы прикладными скриптами и компонентами расширения.
Например, можно в реестре создать параметр **"myfile"**, настроить его значение на некоторый файл, и ссылаться из скрипта на этот файл как **"CashMatic:myfile"**. Реальное полное имя файла доступно для скрипта через вызов функции `CashMatic.Host.GetSpecialURL("myfile")`.

Следует обратить внимание, что специальные URL являются в некотором смысле косвенными (ссылаются на настроечные параметры, которые ссылаются на данные), поэтому сами по себе они не являются файловыми путями в общепринятом понимании - они не могут быть относительными, и нельзя задать путь относительно специального URL. Например, адрес **"CashMatic:data\file"** зависит от значения настроечного параметра **"data\file"**, и не зависит от значений **"data"** или **"file"**.

5.4 Протокол "CashMatic:"

см. также: Модель навигации Специальные URL

NDemia CashMatic KioskBrowser поддерживает в URL-адресах протокол **"CashMatic:"**. Это даёт возможность разработчику приложений ссылаться непосредственно на специальные URL по имени.
При использовании ссылок вида
CashMatic:name
name преобразуется в имя файла или URL-адрес по алгоритму, применяемому для специальных URL (т.е. аналогично вызову метода `GetSpecialURL` объекта `CashMaticHost`).
Ссылки такого типа можно использовать как для переходов между HTML-страницами, так и для любых файловых адресов (например, в HTML-тегах **<a>**, **<img>**, **<iframe>**, **<link>**).
Следует обратить внимание, что специальные URL не являются файловыми путями в общепринятом понимании - они не могут быть относительными, и нельзя задать путь относительно специального URL.
Префикс протокола (URL-схема) **"CashMatic:"** в URL-адресах определяется без различения верхнего-нижнего регистра, однако для определённости рекомендуется именно указанное написание.
URL-протокол **"CashMatic:"** используется в адресах переходов HTML в Примере платёжного интерфейса.

Совместимость: URL-протокол **"CashMatic:"** поддерживается в **NDemia CashMatic**, начиная с версии **2.6.2**

6 Модель обработки отказов

см. также:	Модель навигации	Автоматические переходы	Платёжный сеанс
	Страница неисправности	Контроль URL-адресов переходов	Отложенные отказы

Пользовательский интерфейс платёжных терминалов имеет следующие особенности, отличающие его от обычных веб-сайтов:

1. Обычные пользователи не могут и не должны понимать и решать какие-либо возникающие проблемы в работе аппаратного и программного обеспечения. Проблема, если она возникает, может быть либо проигнорирована (обойдена каким-либо способом), либо пользователю должно быть сообщено о невозможности работы.
2. Даже специально подготовленные пользователи с особым допуском (обслуживающий персонал) работают с программным обеспечением в полевых условиях - без рабочего места, чаще всего без клавиатуры, нередко вообще без каких-либо дополнительных средств. Поэтому, если возникают какие-либо проблемы, то эти проблемы должны диагностироваться и обрабатываться по возможности автоматически.

Исходя из этого, любой отказ аппаратного или программного обеспечения по возможности не должен считаться фатальным с точки зрения продолжения работы программы. Например, при отсутствии бумаги в чековом принтере терминал может уйти в положение "Терминал временно не работает", однако состояние принтера при этом продолжает периодически отслеживаться, и при появлении бумаги терминал должен восстановиться в нормальный режим самостоятельно, не требуя каких-либо дополнительных действий от пользователя.
Каждый возникающий отказ имеет идентифицирующее его имя, которое определяется в зависимости от причин и характера возникающей проблемы - см. Виды отказов терминала.
NDemia CashMatic KioskBrowser поддерживает внутренний список текущих отказов. При нормальной работе этот список пуст, при возникновении (возбуждении) отказа имя отказа добавляется в список, при снятии проблемы имя отказа удаляется из списка.
В зависимости от уровня серьёзности проблемы отказ может потребовать либо некоторых дополнительных программных действий, либо прекращения нормальной работы терминала, т.е. потребуется переход на страницу неисправности - см. Уровни серьёзности отказов, Автоматический переход на страницу неисправности.
В зависимости от текущего момента работы пользовательского интерфейса отказ может быть либо обработан немедленно, либо обработка должна быть отложена до завершения некоторых действий пользователя - см. Отложенные отказы.
Прикладной программист (разработчик прикладных скриптов и компонентов расширения) может предусмотреть собственную обработку для определённых видов отказов (см Обработка отказов терминала), добавить собственные виды отказов (см. метод `SetFailState` объекта `CashMaticTerminal`), а также изменить общий порядок обработки отказов в некоторых особых случаях (см. Дополнительные условия обработки отказов).

6.1 Виды отказов терминала

см. также:	Модель обработки отказов	Уровни серьёзности отказов	Обработка отказов терминала
	Объект CashMaticCash	Объект CashMaticPrint	Объект CashMaticTerminal

Каждый возникающий отказ имеет идентифицирующее его имя, которое определяется в зависимости от причин и характера возникающей проблемы.
В текущей версии **NDemia CashMatic KioskBrowser** определены следующие виды отказов:

- `BillValidator` - общий отказ купюроприёмника;
- `BillCounterOverflow` - переполнение купюроприёмника по программному счётчику (см. `CashMaticCash`);
- `Browser` - общий отказ навигации браузера;
- `Printer` - общий отказ принтера;
- `PaperEnd` - конец бумаги в принтере;
- `NearPaperEnd` - приближение конца бумаги в принтере.

Кроме перечисленных имён отказов, прикладные скрипты и компоненты расширения могут возбуждать собственные "пользовательские" именованные отказы, см. метод `SetFailState` объекта `CashMaticTerminal`. Указанный метод позволяет принудительно явно устанавливать состояние любого именованного отказа, как предопределённого, так и "пользовательского".

> ***Примечание:*** *Крайне не рекомендуется прикладному программисту без существенной необходимости принудительно устанавливать состояния предопределённых отказов, поскольку это может нарушить общую логику работы* ***NDemia CashMatic KioskBrowser****.*

Пример:
```
CashMatic.Terminal.SetFailState("MyFailureName", true);
        //возбуждение "пользовательского" отказа "MyFailureName"
```
В данном случае **"MyFailureName"** - произвольное имя отказа, заданное прикладным программистом. Никаких других деклараций для введения нового имени отказа не требуется. Компоненты расширения получат оповещение о возникновении отказа **"MyFailureName"** через вызовы метода `OnFailure` интерфейса `ICashMaticTerminalExtender` - см. Обработка отказов.
Прикладной программист может повысить уровень серьёзности отказа **"MyFailureName"** до ошибки, требующей автоматического перехода на страницу неисправности - см. Уровни серьёзности отказов:
```
CashMatic.Host.AddErrorName("MyFailureName");
        //это должно быть сделано на начальном этапе работы скрипта

// ... исполняется прикладной скрипт

CashMatic.Terminal.SetFailState("MyFailureName", true);
        //возбуждение "пользовательского" отказа "MyFailureName"
        //      приведёт к автоматическому переходу на страницу неисправности
        //      (кроме случая отложенного отказа).
```

6.2 Уровни серьёзности отказов

см. также:	Модель обработки отказов	Виды отказов терминала	Обработка отказов терминала
	Отложенные отказы	Объект CashMaticHost	Страница неисправности

Реакция **NDemia CashMatic KioskBrowser** на отказ зависит от уровня серьёзности отказа. Серьёзные отказы называются ошибками. Если возбуждается отказ уровня ошибки, то пользовательский интерфейс принудительно переключается на страницу неисправности (см. Автоматический переход на страницу неисправности), в остальных случаях пользователю ничего не сообщается. Во всех случаях возбуждение отказа регистрируется в лог-файле (с соответствующими комментариями), и оповещаются компоненты расширения (см. Обработка отказов терминала).
Модель обработки отказов терминала **NDemia CashMatic** является обобщённой как для предопределённых отказов, так и для "пользовательских" отказов, созданных прикладным программистом (см. Виды отказов терминала).
После запуска **NDemia CashMatic KioskBrowser** по умолчанию ошибками считаются отказы `BillValidator`, `Browser`, `Printer`, `PaperEnd`.
Прикладные скрипты и компоненты расширения могут добавлять в этот список другие имена отказов (в том числе свои собственные) - см. метод `AddErrorName` объекта `CashMaticHost`. В текущей версии возможно только повышение отказа до уровня ошибки - это как бы начальная настройка всего приложения, не изменяющаяся по ходу работы или в зависимости от каких-либо условий.

6.3 Автоматический переход на страницу неисправности

см. также:	Модель обработки отказов	Специальные URL	Отложенные отказы
	Уровни серьёзности отказов	Обработка отказов терминала	Объект CashMaticTerminal

Когда возбуждается отказ уровня ошибки, пользовательский интерфейс автоматически переходит на страницу неисправности (см. Виды отказов, Уровни серьёзности отказов).
Адрес страницы неисправности определяется как специальный URL **"failure"** (**"CashMatic:failure"**) - см. Специальные URL.
Рекомендуется всегда делать страницу неисправности локальной на терминале, чтобы отказ связи с сервером не ломал всё окончательно.
В некоторых случаях ошибка (серъёзный отказ уровня ошибки) не приводит к автоматическому переходу на страницу неисправности - см. Отложенные отказы.

> ***Примечание:*** *В текущей версии при возникновении ошибки в состоянии инкассации (см. Контроль URL-адресов переходов) переход на страницу неисправности запрещён (но будет автоматически выполнен после восстановления купюроприёмника, если ошибка останется действующей). В будущих версиях это поведение может быть изменено.*

Автоматический переход на страницу неисправности выполняется только в момент возбуждения отказа уровня ошибки, если других действующих ошибок нет. Если по какой-либо причине переход в этот момент не произошёл (например, см. Отложенные отказы), то автоматически он не произойдёт и в дальнейшем, даже если возникнут новые ошибки - пока все ошибки не будут сняты. Однако контроль допустимости переходов остаётся в силе (см. Контроль URL-адресов переходов), таким образом, когда отказ перестанет быть отложенным, страница неисправности станет единственным допустимым адресом перехода. В этом случае переход на страницу неисправности должен выполняться прикладным скриптом, отвечающим за сценарий платёжного сеанса (см. Подключение прикладных скриптов, Сеанс приёма платежа) - немедленно после снятия условия отложенного отказа (в текущей версии это означает немедленно после сброса платёжного сеанса).

Когда какой-либо отказ снимается, если текущим положением терминала была страница неисправности и ни один из оставшихся отказов не является ошибкой, то происходит восстановление нормальной работоспособности терминала, поэтому пользовательский интерфейс автоматически переходит на стартовую страницу.
Адрес стартовой страницы определяется как специальный URL **"start"** (**"CashMatic:start"**) - см. Специальные URL, Стартовая страница.

6.4 Отложенные отказы

см. также:	Модель обработки отказов	Специальные URL	Страница неисправности
	Уровни серьёзности отказов	Обработка отказов терминала	Платёжный сеанс

Когда возбуждается отказ уровня ошибки, пользовательский интерфейс автоматически должен переходить на страницу неисправности (см. Виды отказов, Уровни серьёзности отказов, Автоматический переход на страницу неисправности).
Однако в некоторых случаях переход на страницу неисправности может быть отменён - возникает т.н. отложенный отказ (состояние отложенного отказа).
В текущей версии **NDemia CashMatic KioskBrowser** возможны следующие случаи отложенного отказа:

1. Отказ произошёл при наличии в текущем сеансе приёма платежа ненулевой суммы принятых денег и наличии принтера и бумаги - в этом случае переход на страницу неисправности не производится до конца (точнее, до сброса) платёжного сеанса (см. Сеанс приёма платежа). Смысл этого в том, чтобы дать возможность прикладному скрипту завершить диалог с пользователем и обработать платёж до перехода на страницу неисправности. Если отказ остаётся действующим после сброса сеанса приёма платежа, то переход на страницу неисправности не делается автоматически (в будущих версиях это может быть изменено). В текущей версии переход должен делаться прикладным скриптом или другим программным обеспечением, отвечающим за сценарий платёжного сеанса (см. также Подключение прикладных скриптов, Компоненты расширения, Сеанс приёма платежа, Автоматический переход на страницу неисправности).

2. Отказ произошёл в состоянии инкассации (т.е. при снятом/удалённом стекере (кассе) купюроприёмника) - в этом случае переход на страницу неисправности не производится до восстановления купюроприёмника (при снятом стекере купюроприёмника запрещены вообще все переходы, кроме страницы инкассации - см. Контроль URL-адресов переходов). Смысл этого в том, что снятие стекера купюроприёмника указывает на присутствие пользователя с особым статусом (инкассатора или технического специалиста), поэтому следует дать пользователю выполнить нужные ему операции, однако платёж принимать нельзя. В текущей версии, если отказ остаётся действующим после восстановления купюроприёмника, то переход на страницу неисправности делается автоматически (в будущих версиях это поведение может быть изменено).

3. Отказ произошёл, когда пользовательский интерфейс был занят модальным диалогом (таким, как javascript-функция alert()) - в этом случае автоматический переход на страницу неисправности не выполняется до закрытия модального диалога (до освобождения пользовательского интерфейса). В текущей версии, если отказ остаётся действующим после закрытия диалога, то переход на страницу неисправности делается автоматически (как и наоборот: если модальный диалог был показан на странице неисправности, и во время диалога все отказы были устранены, т.е. требуется переход на стартовую страницу — то реально переход будет выполнен после закрытия диалога).

> **Совместимость:** автоматические переходы после закрытия модальных диалогов поддерживаются в **NDemia CashMatic**, начиная с версии **2.7.2**

Обратите внимание: не каждый отказ считается ошибкой (см. Уровни серьёзности отказов). Если отказ не считается ошибкой, то он не требует перехода на страницу неисправности и не запрещает переходов на любые URL-адреса. Если отказ считается ошибкой, то он требует перехода на страницу неисправности, переход выполняется либо автоматически, либо, если отказ отложен по некоторым условиям, то переход должен быть выполнен прикладным скриптом или компонентом расширения при снятии этих условий.

Если отказ был отложен по причине ненулевой суммы принятых денег, то условия изменятся после сброса платёжного сеанса (сумма принятых денег станет нулевой). Прикладной скрипт должен проверить, не требуется ли переход на страницу неисправности, и при необходимости выполнить этот переход (переходы на все остальные адреса будут запрещены до снятия отказа).

Пример:

```
CashMatic.Session.Reset();     //сброс платежного сеанса
if(CashMatic.Terminal.Failure) //истина, если в программе зафиксирован отказ,
                               //требующий перехода на страницу неисправности
       //при ошибках переходим на страницу неисправности
       navigate("CashMatic:failure");
```

Прикладной программист имеет возможность некоторым образом изменять контроль и автоматизацию переходов - см. Дополнительные условия обработки отказов.

Обработка отложенных отказов имеет следующую особенность: если восстановилось нормальное состояние терминала (все ошибки устранились), а переход на страницу неисправности (**"CashMatic:failure"**) к этому моменту так и не произошёл, то не произойдёт и автоматический переход на стартовую страницу (**"CashMatic:start"**), поскольку уже нет смысла сбрасывать контекст платежа (см. Автоматический переход на страницу неисправности, Специальные URL, Модель обработки отказов).

Автоматический переход на страницу неисправности выполняется только в момент возбуждения отказа уровня ошибки, если других действующих ошибок нет. Если по какой-либо причине переход в этот момент не произошёл, то автоматически он не произойдёт и в дальнейшем, даже если возникнут новые ошибки - пока все ошибки не будут сняты.

6.5 Обработка отказов терминала

см. также:	Модель обработки отказов	Виды отказов	Уровни серьёзности отказов
	Компоненты расширения	Объект CashMaticTerminal	Интерфейс ICashMaticTerminalExtender

Обработка отказов терминала может выполняться компонентами расширения **NDemia CashMatic**, регистрирующими обработчики с интерфейсом `ICashMaticTerminalExtender` (см. объект `CashMaticTerminal`). При возникновении или снятии какого-либо отказа компоненты расширения получают соответствующее оповещение через вызов `ICashMaticTerminalExtender::OnFailure`. Указанный метод вызывается в каждом из установленных компонентов (порядок в общем случае не определён).
Компонент расширения или прикладной скрипт могут принудительно установить или снять любой именованный отказ - см. метод `SetFailState` объекта `CashMaticTerminal`.

> ***Примечание:*** *Крайне не рекомендуется прикладному программисту без существенной необходимости принудительно устанавливать состояния предопределённых отказов, поскольку это может нарушить общую логику работы* ***NDemia CashMatic KioskBrowser****.*

При вызове `SetFailState` все компоненты оповещаются так же, как и при реальном ("физическом") отказе или снятии отказа. Следует иметь в виду, что вызов `SetFailState` внутри вызова `ICashMaticTerminalExtender::OnFailure` приводит к рекурсии (все компоненты снова получают оповещение). Такая ситуация нежелательна, поскольку прикладной программист в общем случае не знает, какие ещё компоненты могут быть установлены на терминале, и как они реагируют на тот или иной отказ (хотя в целом рекурсии `OnFailure` в текущей версии разрешены).
Уровень серьёзности отказов не имеет значения с точки зрения вызовов `SetFailState` и `OnFailure`.
При обработке серьёзных отказов (ошибок) автоматические переходы на страницу неисправности и на стартовую страницу выполняются после соответствующих оповещений `OnFailure`.
При отложенном отказе компоненты расширения и прикладные скрипты могут проверить необходимость перехода на страницу неисправности через опрос свойства `Failure` объекта `CashMaticTerminal`.

6.6 Дополнительные условия обработки отказов

см. также: Модель обработки отказов | Контроль URL-адресов переходов | Отложенные отказы
Модель навигации | Автоматические переходы | Платёжный сеанс

Прикладной программист имеет возможность отключить контроль URL-адресов переходов, главным образом это сделано для расширенной обработки ошибок, чтобы предотвратить блокирование переходов пользовательского интерфейса в состоянии отложенного отказа - см. параметр **"Try Start Anyway"** в разделе Контроль URL-адресов переходов.
Кроме того, прикладной программист может отключить автоматический переход на страницу неисправности при одновременном наличии принятых в текущем платёжном сеансе денег и невозможности напечатать чек из-за проблем принтера (неисправность, отсутствие принтера, отсутствие бумаги) - см. Отложенные отказы
раздел реестра
HKEY_CURRENT_USER\Software\NDemia\CashMatic
параметр
"Try Print Anyway"
(тип **REG_DWORD**),
ненулевое значение отключает автоматический переход.

> ***Примечание:*** *Параметр **"Try Print Anyway"** отключает не автоматические переходы вообще, а только автоматический переход на страницу ошибки при достаточно определённых условиях: в текущем платёжном сеансе были приняты деньги, но чек напечатать предположительно не удастся.*

Рекомендуемое использование: компонент расширения, реализующий собственную расширенную обработку завершения платежа (например, зачисление на сервер), может временно отключать автоматический переход по инициативе браузера.
Не рекомендуется постоянное отключение автоматического перехода, поскольку он встроен в программу с целью надёжного предотвращения возможности работы терминала в неисправном состоянии (как своего рода предохранитель).

Установка флага **"Try Print Anyway"** отключает автоматический переход, но не отключает обработку отказов и контроль URL-адресов переходов. Несмотря на то, что автоматический переход на страницу неисправности в режиме **"Try Print Anyway"** не выполняется, внутренне состояние терминала переходит в режим ошибки - возникает отложенный отказ.

В текущей реализации, если автоматический переход на страницу неисправности был отключен в момент возникновения отказа, то автоматический переход затем не будет выполняться и при снятии **"Try Print Anyway"**, до тех пор, пока все отказы не будут устранены (т.е. пока общее состояние терминала не вернётся к норме). Другими словами, если терминал не отреагировал на ошибку (из-за временного отключения этой реакции), то затем он не будет реагировать ни на эту, ни на какие другие последующие ошибки, пока все ошибки не будут сняты. Это связано с тем, что автоматический переход выполняется только при изменении общего состояния терминала "нет ошибок->есть ошибки", это состояние определяется внутренними условиями и не связано с адресом текущей HTML-страницы.

Совместимость: параметр **"Try Print Anyway"** поддерживается в **NDemia CashMatic**, начиная с версии **2.7.0**

7 Взаимодействие с купюроприёмником

Безусловно, основным назначением платёжного терминала является приём наличных денег. Поэтому взаимодействие с купюроприёмником является одной из главных задач, решаемых с помощью **NDemia CashMatic KioskBrowser**.
Объект `CashMaticCash` отвечает за купюроприёмник и является основным средством организации приёма купюр от пользователя.

Сеанс купюроприёмника (см. объект `CashMaticCashSession`) - это процесс, в ходе которого купюроприёмник принимает купюры от пользователя. С точки зрения общего сценария платежа (сценария работы пользователя) приём купюр часто удобно рассматривать как одно целостное событие или состояние: "пользователь вносит деньги". На самом деле за время этого состояния происходит ряд переключений - сеанс купюроприёмника состоит из одного и более циклов включения-выключения приёма купюр на купюроприёмнике, причём каждый цикл может отличаться от других диапазоном принимаемых купюр, таймаутом и т.п.
NDemia CashMatic сообщает прикладному скрипту о различных событиях, связанных с купюроприёмником и соответствующих различным изменениям параметров, успешному или неуспешному приёму купюр и т.п. - см. Примеры обработчиков событий сеанса купюроприёмника.

Основным режимом использования купюроприёмника в **NDemia CashMatic** является приём купюр, кроме этого поддерживается дополнительный режим - проверка купюр (с вовзратом купюр пользователю, без складирования), см. Режимы проверки купюр и приёма купюр.

NDemia CashMatic предоставляет прикладному программисту ряд объектов, которые можно использовать для получения информации о принятых купюрах:

- `CashMaticCurrency` - денежное значение с включением кода валюты, безотносительно количества купюр;
- `CashMaticBill` - денежный номинал одной купюры;
- `CashMaticBillCounter` - суммарное значение некоторого количества купюр одного достоинства;
- `CashMaticCashCounter` - сумма купюр произвольного (одного или разного) достоинства.

NDemia CashMatic имеет собственный счётчик принятых купюр, значение которого хранится в системном реестре - см. Счётчик купюр.

Прикладной скрипт имеет возможность узнать протокол управления купюроприёмника, а для купюроприёмника, работающего по протоколу CCNET, получить идентификацию оборудования: версию прошивки, серийный номер и уникальный код купюроприёмника - см. Примеры опроса идентификации купюроприёмника.

Аппаратно-программный интерфейс купюроприёмника сообщает компьютеру не номиналы (денежные достоинства), а типы купюр - условные номера, которые затем программно сопоставляются денежным номиналам - см. Соответствие типов купюр номиналам для российских рублей. При необходимости преобразовать тип принятой/проверенной купюры, полученный на интерфейсе купюроприёмника, в номинал (денежное достоинство) купюры, прикладной программист может воспользоваться методом `BillTypeToMoney()` объекта `CashMaticCash`. Однако использование типов купюр в прикладных скриптах на текущий момент эволюции **NDemia CashMatic** является устаревшей методикой - для программиста, работающего в объектной модели **NDemia CashMatic**, нет необходимости знать и использовать внутренние условные номера типов купюр (такая необходимость на данный момент сохраняется для программиста, пользующегося дополнительными средствами типа команды `SIMCASH`, кроме того, тест купюроприёмника в Панели управления **NDemia CashMatic** сообщает именно тип купюры в чистом виде, без преобразования в номинал).

Кроме объектов, доступных для прикладных скриптов, программист может воспользоваться некоторыми дополнительными средствами. По задачам взаимодействия с купюроприёмником могут использоваться следующие дополнительные команды: `BVSIMEVENT`, `BVSTATMON`, `CONFIG`, `GETCASH`, `SIMCASH`. Короткие описания команд, которые могут понадобиться для отладки клиентских приложений, приведены в разделе "Дополнительные средства. Сокращённые сведения".

7.1 Обзоры

7.1.1 Счётчик купюр

Купюроприёмники, поддерживаемые в **NDemia CashMatic KioskBrowser**, не имеют встроенного счётчика купюр, но имеют сигнал переполнения и сигнал снятия (удаления) стекера (кассы).
NDemia CashMatic имеет собственный счётчик принятых купюр, значение которого хранится в системном реестре - см. свойство `BillCounter` объекта `CashMaticCash`.
Однако следует понимать, что это только программный счётчик, он учитывает только купюры, которые были обработаны через **NDemia CashMatic** и не учитывает реального физического состояния кассы. Инкассацией считается снятие стекера, при этом программный счётчик сбрасывается. Здесь тоже следует понимать, что реально нет возможности узнать, был ли стекер в результате его снятия разгружен или заменён на пустой.
В Панели управления **NDemia CashMatic** (вкладка "Терминал") может быть задано предельное значение счётчика купюр, при достижении этого значения возбуждается отказ `BillCounterOverflow` (см. Модель обработки отказов). В текущей версии **NDemia CashMatic** проверка переполнения делается при вызове метода `CheckBillCounter()` объекта `CashMaticCash`.
При инкассации может выполняться автоматическая печать отчёта с гашением на фискальном регистраторе (Z-отчёта) - см. свойство `AutoZReport` объекта `CashMaticCash`.

7.1.2 Режимы проверки купюр и приёма купюр

Купюроприёмник используется для приёма наличных денег (купюр) от пользователя. Кроме этого, в **NDemia CashMatic** поддерживается дополнительный режим - проверка купюр.
В режиме приёма купюр купюроприёмник определяет номинал (денежное достоинство) купюры, после чего складирует её в стекер (кассу) и сообщает об этом прикладной программе.
В режиме проверки купюр купюроприёмник так же определяет номинал купюры, после чего возвращает её пользователю, сообщая прикладной программе распознанный номинал.
Режим проверки можно использовать, например, для проверки исправности купюроприёмника (или можно предложить пользователю терминала некоторый дополнительный сервис).
Режим проверки задействуется при присваивании истинного значения свойству `Detect` объекта `CashMaticCashSession`. По умолчанию свойство `Detect` имеет ложное значение, поэтому без дополнительных указаний программиста купюроприёмник всегда работает в режиме приёма купюр.
Прикладная программа защищена от неправильной интерпретации результатов - режимы проверки и приёма разделены по обработчикам событий, счётчикам купюр и т.п. (за исключением событий и счётчиков, значение которых не зависит от выбора режима, например таких, как событие `OnTimeout` или свойство `RejectCount` объекта `CashMaticCashSession`)

Свойства и методы объекта `CashMaticCashSession`, действующие только в режиме приёма купюр (когда `Detect==false`):

- `Accepted` - принимаемая купюра, номинал;
- `BillCount` - счётчик принятых купюр;
- `BillTotal` - счётчик принятых наличных в денежном выражении;
- `GetBillCountByDenom()` - количество принятых купюр указанного номинала;
- `GetBillCounters()` - коллекция счётчиков принятых купюр по номиналам;
- `LastAccepted` - последняя принятая купюра, номинал;
- `OnAccept` - событие приёма купюры.

Свойства и методы объекта `CashMaticCashSession`, действующие только в режиме проверки купюр (когда `Detect==true`):

- `DetectCount` - счётчик проверенных купюр;
- `Detected` - проверяемая купюра, номинал;
- `LastDetected` - последняя проверенная купюра;
- `OnDetect` - событие проверки купюры.

7.2 Объекты

CashMaticBill
CashMaticBillCounter
CashMaticCash
CashMaticCashCounter
CashMaticCashDeviceIdentification
CashMaticCashSession
CashMaticCurrency
CashMaticItems
CashMaticRequest
CashMaticResult

7.2.1 Объект CashMaticBill, свойства

см. также:	CashMaticBillCounter	CashMaticCash	CashMaticCashCounter
	CashMaticCashSession	CashMaticCashSessionEvents	CashMaticCurrency

Объект CashMaticBill является основным описанием купюры в **NDemia CashMatic**, включающим:

- номинал (денежное значение) купюры, как свойство Value;
- код валюты купюры по стандарту **ISO 4217**, как свойство Code;
- тип купюры в обозначении интерфейса купюроприёмника, как свойство Type (использование типов купюр в прикладных скриптах на текущий момент эволюции **NDemia CashMatic** является устаревшей методикой - для программиста, работающего в объектной модели **NDemia CashMatic**, нет необходимости знать и использовать внутренние условные номера типов купюр).

Наличие свойства Value позволяет использовать объект в скриптах как значение свойства Value.
В текущей версии **NDemia CashMatic** объект CashMaticBill используется как значение или параметр в ряде свойств и методов объекта CashMaticCashSession и интерфейса CashMaticCashSessionEvents.

Родственными по смыслу объектами являются:

- CashMaticBillCounter - суммарное значение некоторого количества купюр одного достоинства;
- CashMaticCashCounter - сумма купюр произвольного (одного или разного) достоинства;
- CashMaticCurrency - денежное значение с включением кода валюты, безотносительно количества купюр.

Имя	Тип результата	Тип обращения	Назначение
Code	Строка	*Свойство, только чтение*	код валюты по стандарту ISO 4217.
Type	Число	*Свойство, только чтение*	тип купюры в обозначении интерфейса купюроприёмника (см. соответствие типов купюр номиналам для российских рублей).
Value	Число	*Свойство, только чтение*	номинал (денежное значение) купюры. *(наличие поля **Value** позволяет в скриптах использовать объект **CashMaticBill** как число, равное номиналу купюры)*

Совместимость: объект CashMaticBill поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.2 Объект CashMaticBillCounter, свойства

см. также:	CashMaticBill	CashMaticCash
	CashMaticCashCounter	CashMaticCurrency

Объект CashMaticBillCounter описывает сумму (набор, группу, блок, пачку) купюр одного (одинакового) номинала (денежного достоинства).
Свойство Bill возвращает единичную купюру ("образец"), соответствующую входящим в данную сумму, как объект CashMaticCurrency. Данная купюра имеет ненулевое денежное значение даже при нулевом количестве купюр - например, если рассматривать ситуацию "0 купюр по 1000 рублей", то значением свойства Bill будет 1000 RUB.
Количество купюр доступно как свойство Count или как свойство Value (эквивалентно).
Наличие свойства Value позволяет использовать объект в скриптах как значение свойства Value.
Свойство Total возвращает сумму купюр в денежном выражении, как объект CashMaticCurrency.
В текущей версии объект CashMaticBillCounter создаётся:

- как возвращаемое значение метода GetBillCountByDenom() объекта CashMaticCashSession;
- как элемент коллекции CashMaticItems, возвращаемой методом GetBillCounters() объекта CashMaticCashSession.

Свойства объекта CashMaticBillCounter доступны только для чтения, поскольку он выражает некоторое зафиксированное состояние суммы купюр, которое не подлежит программным изменениям.
Родственными по смыслу объектами являются:

- CashMaticBill - денежный номинал одной купюры;
- CashMaticCashCounter - сумма купюр произвольного (одного или разного) достоинства;
- CashMaticCurrency - денежное значение с включением кода валюты, безотносительно количества купюр.

Имя	Тип результата	Тип обращения	Назначение
Bill	объект **CashMaticBill**	*Свойство, только чтение*	единичная купюра ("образец"), соответствующая данному счётчику купюр (может иметь ненулевое значение даже при нулевом количестве купюр, например "0 купюр по 1000 рублей").
Count	Число	*Свойство, только чтение*	количество купюр.
Total	Объект **CashMaticCurrency**, число	*Свойство, только чтение*	сумма купюр в денежном выражении. *Возвращаемое значение может использоваться как численное, см. свойство **CashMaticCurrency.Value**.*
Value	Число	*Свойство, только чтение*	количество купюр (эквивалентно значению **Count**). *(наличие поля **Value** позволяет в скриптах использовать объект **CashMaticBillCounter** как число, равное количеству купюр)*

Совместимость: объект CashMaticBillCounter поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.3 Объект CashMaticCash (CashMatic.Cash), свойства и методы

см. также:	CashMaticCashSession	CashMaticCashDeviceIdentification
	CashMaticCurrency	CashMaticRequest

Объект CashMaticCash (доступен из прикладного скрипта как свойство CashMatic.Cash) является основным объектом взаимодействия с купюроприёмником в объектной модели **NDemia CashMatic**.
Объект CashMaticCash является в некотором смысле "стационарным" объектом, связанным с оборудованием, а не с текущим платежом и другими действиями пользователя.
Непосредственно приём или проверка купюр выполняются в сеансовом режиме, за организацию таких сеансов отвечают объекты CashMaticCashSession, которые создаются в результате вызова метода CreateSession() объекта CashMaticCash. Через свойство ActiveSession объекта CashMaticCash можно получить ссылку на текущий активный сеанс купюроприёмника (под активностью в данном случае понимается непосредственное использование купюроприёмника для приёма или проверки купюр).
Объект CashMaticCash предоставляет доступ к программному счётчику купюр купюроприёмника - см. свойство BillCounter. В текущей версии предоставляется только информация о количестве купюр (без подсчёта денежных значений и счётчиков по отдельным типам купюр), в дальнейшем предполагается предоставлять информацию на уровне детализации, аналогичном методу GetBillCounters() объекта CashMaticCashSession.
Для проверки переполнения купюроприёмника по программному счётчику купюр прикладной скрипт должен вызывать метод CheckBillCounter(). По результату этой проверки возвращается логическое значение (истина соответствует отсутствию переполнения) и выставляется состояние отказа BillValidator (см. Модель обработки отказов)

Объект CashMaticCash поддерживает методы опроса идентификации подключенного оборудования: GetDeviceProtocol() и GetDeviceIdentification() (см. Примеры опроса идентификации купюроприёмника).
Метод GetDeviceProtocol() опрашивает настройку протокола управления купюроприёмника по параметрам, выставленным в Панели управления **NDemia CashMatic**, на вкладке "Купюроприёмник" (т.е. это не реальный опрос физического устройства, а получение выставленной вручную программной настройки).
Метод GetDeviceIdentification() опрашивает купюроприёмник и создаёт объект CashMaticCashDeviceIdentification - только если купюроприёмник реально подключен и поддерживает протокол CCNET.
Методы GetDeviceProtocol() и GetDeviceIdentification() могут выполняться как синхронно (т.е. с возвратом непосредственного результата), так и асинхронно (через запрос с обработкой результата, отложенной до готовности ответа).
При необходимости прикладной скрипт может прервать все незавершённые асинхронные запросы вызовом метода CancelDeviceRequests().

Свойство AutoZReport объекта CashMaticCash является постоянно действующей настройкой **NDemia CashMatic**. Если это свойство имеет истинное значение, то при выполнении инкассации автоматически вызывается печать отчёта с гашением (вызывается метод ZReport() объекта CashMaticPrint). Присвоенное значение AutoZReport сохраняется в системном реестре Windows.
При необходимости преобразовать тип принятой/проверенной купюры, полученный на интерфейсе купюроприёмника, в номинал (денежное достоинство) купюры, прикладной программист может воспользоваться методом BillTypeToMoney() (использование типов купюр в прикладных скриптах на текущий момент эволюции **NDemia CashMatic** является устаревшей методикой - для программиста, работающего в объектной модели **NDemia CashMatic**, нет необходимости знать и использовать внутренние условные номера типов купюр).
Свойство DefaultCurrency объекта CashMaticCash возвращает код валюты, действующей по умолчанию в данной установке **NDemia CashMatic**. Валюта по умолчанию возвращается в виде объекта CashMaticCurrency с нулевым денежным значением, код валюты доступен через свойство Code этого объекта. В текущей версии **NDemia CashMatic валютой** по умолчанию является российский рубль -

код **"RUB"**.

Метод `ResetDevice()` выполняет командный (программный) сброс купюроприёмника (купюроприёмнику через протокол управления подаётся команда сброса). Данный метод носит вспомогательный характер, он может использоваться при попытке восстановления работоспособности купюроприёмника со страницы неисправности терминала.

Имя	Тип результата	Тип обращения	Назначение
ActiveSession	Объект **CashMaticCashSession**, или `null`	*Свойство, чтение/запись*	Активный сеанс купюроприёмника (т.е. сеанс, в котором разрешена вставка купюр в купюроприёмник для проверки или приёма). Прямое присваивание значения свойства разрешено, но не рекомендуется (с точки зрения исходных текстов - управление свойством **Active** объекта **CashMaticCashSession** представляется более понятным и менее "неявно косвенным" действием). Если выполняется прямое присваивание значения свойства **ActiveSession**, то должны быть соблюдены условия возможности изменения свойства **Active** и для текущего (если не `null`), и для нового (если не `null`) сеансов купюроприёмника — см. **CashMaticCashSession** (*). **Совместимость:** поддерживается, начиная с версии **2.6.0**
AutoZReport	Логический	*Свойство, чтение/запись*	Автоматическая печать Z-отчёта при инкассации (удалении стекера купюроприёмника). На данный момент поддерживается только при наличии фискального регистратора. При изменении значение записывается в реестр.
BillCounter	Число	*Свойство, только чтение*	Текущее значение программного счётчика купюр (программный счётчик купюр автоматически сбрасывается при инкассации, т.е. при удалении стекера купюроприёмника) Установленный после удаления купюроприёмник считается пустым, точная информация о его наполнении недоступна (см. Счётчик купюр).
BillTypeToMoney(BillType)	Объект **CashMaticCurrency**	*Метод*	Преобразование типа купюры в денежное значение купюры. Параметр **BillType** - тип купюры (число - внутреннее представление купюры на интерфейсе купюроприёмника, см. Соответствие типов купюр номиналам для российских рублей). *Возвращаемое значение может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticCurrency.Value**.* Для несуществующих/ неизвестных/ неподдерживаемых типов купюр возвращается нулевое значение с пустым кодом валюты (объект **CashMaticCurrency**, свойство **Value** равно нулю, свойство **Code** - пустое значение). **Совместимость:** поддерживается, начиная с версии **2.6.0**
CancelDeviceRequests()	Пустой	*Метод*	Отмена всех асинхронных запросов к устройству (реализованных объектами **CashMaticRequest**). *В текущей версии **NDemia CashMatic** асинхронные запросы к купюроприёмнику создаются методами **GetDeviceProtocol()** и **GetDeviceIdentification()**.* Все незавершённые запросы немедленно завершаются с результатом, соответствующим отмене запроса (см. метод **CancelRequest** объекта **CashMaticRequest**). **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
CheckBillCounter()	Логический	*Метод*	Проверка переполнения купюроприёмника (по программному счётчику - см. свойство **BillCounter**, предел наполнения купюроприёмника задаётся в настройках программы, через Панель управления **NDemia CashMatic**; см. Счётчик купюр). Внутренне вызывает соответствующее изменение состояние отказа **BillCounterOverflow** (см. Модель обработки отказов). Возвращаемое значение: истина - счётчик купюр не достиг уровня переполнения, отказ **BillCounterOverflow** снят; ложь - купюроприёмник переполнен (по программному счётчику купюр), возбуждён отказ **BillCounterOverflow**.
CreateSession()	Объект **CashMaticCashSession**	*Метод*	Создание нового объекта **CashMaticCashSession**, отвечающего за сеанс приёма/проверки наличных через купюроприёмник. **Объект CashMaticCashSession существует вне программной связи с текущим платёжным сеансом (объектом CashMaticSession), поэтому прикладной программист должен явно управлять сеансом купюроприёмника (включать/выключать приём/проверку наличных), в том числе явно завершить сеанс купюроприёмника перед завершением платёжного сеанса пользователя.**
DefaultCurrency	Объект **CashMaticCurrency**	*Свойство, только чтение*	"образец" валюты по умолчанию - объект **CashMaticCurrency** с нулевым денежным значением (свойство **Value** равно нулю), свойство **Code** содержит код валюты по умолчанию (по стандарту ISO 4217). *В текущей версии **NDemia CashMatic** валютой по умолчанию является российский рубль (`"RUB"`).* **Совместимость:** поддерживается, начиная с версии **2.6.0**
GetDeviceIdentification ([ResultHandler])	Объект **CashMaticRequest**, или объект **CashMaticCashDeviceIdentification**	*Метод*	Опрос идентификации купюроприёмника (см. Примеры опроса идентификации купюроприёмника). Если указан параметр **ResultHandler** (объект, поддерживающий интерфейс **CashMaticResultHandler**), то опрос выполняется асинхронным запросом (метод возвращает объект **CashMaticRequest**, по готовности результата вызывается метод **HandleResult** обработчика результата запроса, указанного параметром **ResultHandler**, результатом, передаваемым через свойство **Data** объекта **CashMaticResult**, является объект **CashMaticCashDeviceIdentification** (•). Если параметр **ResultHandler** не указан, то опрос идентификации выполняется непосредственно при вызове метода **GetDeviceIdentification** (синхронно), метод возвращает объект **CashMaticCashDeviceIdentification**. Опрос идентификации в режиме приёма или проверки купюр с большой вероятностью закончится неудачей (••). *Идентификация оборудования поддерживается только для купюроприёмников, работающих по протоколу CCNET, поэтому рекомендуется перед вызовом **GetDeviceIdentification** получить протокол устройства через вызов метода **GetDeviceProtocol**.* **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
GetDeviceProtocol ([ResultHandler])	Объект **CashMaticRequest**, или строка	*Метод*	Опрос протокола управления купюроприёмника (см. Примеры опроса идентификации купюроприёмника). Если указан параметр **ResultHandler** (объект, поддерживающий интерфейс **CashMaticResultHandler**), то опрос выполняется асинхронным запросом (метод возвращает объект **CashMaticRequest**, по готовности результата вызывается метод **HandleResult** обработчика результата запроса, указанного параметром **ResultHandler**, результатом, передаваемым через свойство **Data** объекта **CashMaticResult**, является строка, идентифицирующая протокол управления купюроприёмника (•). Если параметр **ResultHandler** не указан, то опрос протокола выполняется непосредственно при вызове метода **GetDeviceProtocol** (синхронно), метод возвращает строку, идентифицирующую протокол управления купюроприёмника. *Настройка протокола управления купюроприёмника выполняется в Панели управления **NDemia CashMatic**, на вкладке "Купюроприёмник".* Поддерживаются следующие идентификаторы протоколов: `"CCNET"` - купюроприёмники производства CashCode (http://cashcode.com); `"ICT004"` - купюроприёмники производства International Currency Technologies (http://www.ictgroup.com.tw). **Совместимость:** поддерживается, начиная с версии **2.6.0**
ResetDevice()	Пустой	*Метод*	Командный (программный) сброс купюроприёмника. Может использоваться как дополнительная мера при попытке восстановления работоспособности со страницы неисправности.

(•) Результат асинхронного вызова рассматривается как положительный, только если код результата (свойство `Code` объекта CashMaticResult) имеет нулевое значение, иначе остальные свойства CashMaticResult не рассматриваются. Другие возможные значения для кода результата:

- 1 — таймаут, данные не получены за отведённое для этого время;
- 2 — оффлайн, нет связи с купюроприёмником;
- 4 — отмена, запрос прерван (например, из-за завершения работы терминала).

(••) если во время опроса идентификации устройства выполняется активный сеанс приёма или проверки купюры, то идентификация купюроприёмника будет получена по окончании сеанса приёма или проверки купюры. Ожидание данных ограничено таймаутом (по умолчанию 5 секунд) — если за это время активный сеанс не завершится, то опрос идентификации закончится неудачей (см. Примеры опроса идентификации купюроприёмника).

7.2.4 Объект CashMaticCashCounter, свойства

см. также:	CashMaticBillCounter	CashMaticCash
	CashMaticBill	CashMaticCurrency

Объект CashMaticCashCounter описывает сумму (набор, группу, блок, пачку) купюр произвольного (одного или разного) достоинства.
Количество купюр доступно как свойство Count или как свойство Value (эквивалентно).
Наличие свойства Value позволяет использовать объект в скриптах как значение свойства Value.
Свойство Total возвращает сумму купюр в денежном выражении, как объект CashMaticCurrency.

В текущей версии объект CashMaticCashCounter создаётся только как возвращаемое значение свойства BillCount объекта CashMaticCashSession.

Свойства объекта CashMaticCashCounter доступны только для чтения, поскольку он выражает некоторое зафиксированное состояние суммы купюр, которое не подлежит программным изменениям.

Родственными по смыслу объектами являются:

- CashMaticBill - денежный номинал одной купюры;
- CashMaticBillCounter - суммарное значение некоторого количества купюр одного достоинства;
- CashMaticCurrency - денежное значение с включением кода валюты, безотносительно количества купюр.

Имя	Тип результата	Тип обращения	Назначение
Count	Число	*Свойство, только чтение*	количество купюр.
Total	Объект **CashMaticCurrency**, число	*Свойство, только чтение*	сумма купюр в денежном выражении. *Возвращаемое значение может использоваться как численное, см. свойство **CashMaticCurrency.Value**.*
Value	Число	*Свойство, только чтение*	количество купюр (эквивалентно значению **Count**). *(наличие поля **Value** позволяет в скриптах использовать объект **CashMaticCashCounter** как число, равное количеству купюр)*

Совместимость: объект CashMaticCashCounter поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.5 Объект CashMaticCashDeviceIdentification, свойства

см. также: CashMaticCash CashMaticResult CashMaticResultHandler

Объект CashMaticCashDeviceIdentification предоставляет сведения об идентификации купюроприёмника (только при использовании протокола CCNET).
Для получения этого объекта нужно вызвать метод GetDeviceIdentification объекта CashMaticCash, либо синхронно (без параметров), либо асинхронно, указав в параметре обработчик результата запроса (объект, реализующий интерфейс CashResultHandler) - см. Примеры опроса идентификации купюроприёмника.
При асинхронном вызове по готовности результата будет вызван метод HandleResult интерфейса CashMaticResultHandler. Этот метод получает результат выполнения запроса в виде объекта CashMaticResult. Свойство Data объекта CashMaticResult при успешном завершении запроса будет возвращать объект CashMaticCashDeviceIdentification.

Имя	Тип результата	Тип обращения	Назначение
AssetNumber	Строка	*Свойство, только чтение*	элемент идентификации купюроприёмника "Asset Number" (двоичный код, уникальный для каждого купюроприёмника).
PartNumber	Строка	*Свойство, только чтение*	элемент идентификации купюроприёмника "Part Number" (модель купюроприёмника, точнее версия прошивки).
SerialNumber	Строка	*Свойство, только чтение*	элемент идентификации купюроприёмника "Serial Number" (серийный номер купюроприёмника).

Совместимость: объект CashMaticCashDeviceIdentification поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.6 Объект CashMaticCashSession, свойства и методы

см. также:	CashMaticBill	CashMaticBillCounter	CashMaticCash
	CashMaticCashCounter	CashMaticCashSessionEvents	CashMaticCurrency
	BVSIMEVENT	GETCASH	SIMCASH

Объект `CashMaticCashSession` отвечает за сеанс купюроприёмника.
Сеанс купюроприёмника - это процесс, в ходе которого купюроприёмник принимает купюры от пользователя.
Сеанс купюроприёмника рассматривается отдельно от платёжного сеанса (см. Сеанс приёма платежа).
Обычно сеанс купюроприёмника входит как стадия в платёжный сеанс, однако возможны и другие сценарии (например, когда приём купюр постоянно разрешён).
С точки зрения общего сценария платежа (сценария работы пользователя) приём купюр часто удобно рассматривать как одно целостное событие или состояние: "пользователь вносит деньги". На самом деле за время этого состояния происходит ряд переключений - сеанс купюроприёмника состоит из одного и более циклов включения-выключения приёма купюр на купюроприёмнике, причём каждый цикл может отличаться от других диапазоном принимаемых купюр, таймаутом и т.п.
Объект `CashMaticCashSession` разработан с целью упрощения взаимодействия прикладного скрипта с купюроприёмником, одновременно обеспечивая программиста инструментами гибкого управления процессом приёма купюр и надёжными средствами подсчёта принятых купюр.
Объект `CashMaticCashSession` отвечает за все деньги, принятые от текущего пользователя. Теоретически можно создать одновременно несколько объектов `CashMaticCashSession`, тогда это будет как бы несколько пулов денег - но на практике вряд ли стоит конструировать столь запутанный для пользователя механизм.

Объект `CashMaticCashSession` поддерживает особый режим - проверка купюр (см. Режимы проверки купюр и приёма купюр). В режиме проверки купюры не принимаются, а только детектируются (определяется подлинность и номинал), после чего не складируются, а возвращаются пользователю. Этот режим можно использовать, например, для проверки исправности купюроприёмника (или можно предложить пользователю терминала некоторый дополнительный сервис).

Основным элементом управления работой оборудования по приёму купюр в **NDemia CashMatic** является свойство `Active` объекта `CashMaticCashSession` (логическое значение).

Присваивание `Active=true` включает купюроприёмник на приём/проверку наличных (приём или проверка - в зависимости от значения свойства `Detect`, истинное значение соответствует проверке, ложное - приёму).
Далее приём или проверка наличных будет выполняться до одного из следующих событий:

1. приём купюры (вызывается событие `OnAccept`) или проверка купюры (вызывается событие `OnDetect`);

> ***Примечание:*** *положительный результат ожидания купюры может быть симулирован командой `SIMCASH`, описанной в разделе Дополнительные средства. Эта команда специально предусмотрена для возможности отладки обработки приёма наличных денег в прикладных скриптах и компонентах расширения (плагинах)* ***NDemia CashMatic*** *без использования и даже без подключения купюроприёмника.*

2. превышение таймаута, заданного свойством `Timeout` (вызывается событие `OnTimeout`);
3. отказ купюроприёмника в приёме купюры (вызывается событие `OnReject`);
4. присваивание `Active=false` или вызов метода `Cancel()` (вызывается событие `OnCancel`);
5. отказ из-за нештатного состояния оборудования по решению драйвера купюроприёмника (вызывается событие `OnCancel`);

> ***Примечание:*** *механизм мониторинга состояния купюроприёмника в* ***NDemia CashMatic*** *в общем случае не требует дополнительного программирования или настроек, однако при необходимости прикладной программист имеет возможность вмешаться его в работу. Для этого следует ознакомиться с командами `BVSTATMON`, `CONFIG BillValidatorBadStatusMask`, `GETCASH` (команды описаны в разделе Дополнительные средства). Команда `BVSIMEVENT` может быть использована для отладки обработки различных состояний купюроприёмника в прикладных скриптах и компонентах расширения (плагинах) без физического подключения купюроприёмника.*

6. общий отказ (вызывается событие `OnFailure`);
7. удаление объекта `CashMaticCashSession` (никакие события не вызываются).

После возникновения одного из вышеперечисленных событий свойство `Active` переходит в **`false`**. Если требуется/допускается дальнейшее продолжение приёма купюр, требуется явно установить `Active` обратно в **`true`** в коде обработки соответствующего события.
При каждом изменении свойства `Active` вызывается событие `OnActive`.
Присваивание `Active=false` (или вызов метода `Cancel()`) выключает приём купюр на купюроприёмнике.
Приём купюр на купюроприёмнике нужно обязательно явно выключать до прекращения существования объекта `CashMaticCashSession`, чтобы можно было узнать окончательное количество принятых денег, иначе есть риск потерять последнюю введённую пользователем купюру. Если по какой-либо причине объект будет удалён без явного прекращения приёма купюр, то может оказаться пропущенной последняя вставка купюры пользователем в купюроприёмник, поскольку не сработают обработчики событий купюроприёмника, связанные с `CashMaticCashSesssion`, т.е. последняя купюра окажется вообще нигде не зафиксированной (кроме лог-файлов).

> ***Примечание:*** *прикладному программисту* ***настоятельно рекомендуется*** *в своей разработке принять* ***все возможные меры для явного прекращения приёма купюр в любой ситуации*** *- использовать в javascript обработку* ***`try-catch`****, событие* ***`onunload`****, и т.п. механизмы, обеспечивающие выполнение кода даже в заранее непредусмотренных нештатных ситуациях.*
> ***Следует помнить****: приём купюр должен быть остановлен (***`Active=false`****) до анализа окончательного количества принятых денег, иначе всегда возможен интервал хотя бы в малую долю секунды, за который может поступить сигнал об ещё одной принятой купюре, и она может оказаться нигде не учтённой.*
> *Присваиванию* ***`Active=false`*** *вполне можно доверять, это не просто сброс флага - запускается целый ряд механизмов внутренней синхронизации* ***NDemia CashMatic****, которые однозначно гарантируют, что после выполнения* ***`Active=false`*** *больше не будет принята ни одна купюра.*

Существует возможность ограничить диапазон допустимых номиналов принимаемых купюр (минимальный и максимальный номинал принимаемой купюры), используя свойства `MinBill`, `MaxBill` или методы `SetBillRange()`/`ResetBillRange()`.
Примеры:

- `MinBill=100` - не принимать купюры с номиналом менее 100 рублей.
- `MaxBill=500` - не принимать купюры с номиналом более 500 рублей.
- `SetBillRange(50, 1000)` - принимать купюры только от 50 рублей до 1000 рублей.
- `ResetBillRange()` - снять все предыдущие ограничения, далее принимать все купюры.

При изменении диапазона допустимых номиналов принимаемых купюр вызывается событие `OnBillRange`.

Объект `CashMaticCashSession` предоставляет информацию о номинале принятой или проверенной купюры:

- свойство `Accepted` возвращает принятую купюру (доступно во время выполнения события `OnAccept`);
- свойство `LastAccepted` возвращает последнюю принятую купюру на текущий момент (доступно постоянно);
- свойство `Detected` возвращает проверенную купюру (доступно во время выполнения события `OnDetect`);
- свойство `LastDetected` возвращает последнюю проверенную купюру на текущий момент (доступно постоянно).

Объект CashMaticCashSession предоставляет прикладному программисту несколько способов получения информации о происходящих событиях сеанса купюроприёмника (см. Примеры обработчиков событий сеанса купюроприёмника).
Обработчиком события может быть текстовая строка, которая при вызове события будет выполнена как код javascript в контексте текущего документа.
Обработчиком события может быть функция javascript, которая будет выполняться при вызове события.
Обработчик в виде строки или функции у события может быть только один, назначение нового кода обработки заменяет предыдущий код, если он был.
Кроме того, прикладной скрипт может создать объект-обработчик событий купюроприёмника (см. интерфейс CashMaticCashSessionEvents)
Объектный обработчик событий купюроприёмника устанавливается вызовом метода AddHandler и удаляется вызовом метода RemoveHandler объекта CashMaticCashSession.
Объектных обработчиков событий купюроприёмника может быть создано и установлено одновременно несколько (более одного), они существуют и вызываются независимо друг от друга, а также от простых обработчиков (т.е. от назначенных в виде строки или функции).

Объект CashMaticCashSession отвечает за подсчёт принятых и проверенных купюр.
Прикладной программист может использовать различные уровни детализации счётчиков, в зависимости от своих задач:

- свойство BillCount возвращает количество купюр, принятых в ходе сеанса купюроприёмника (не денежную сумму, а именно количество купюр, проверенные и возвращённые купюры не входят в это значение);
- свойство BillTotal возвращает денежную сумму, принятую в ходе сеанса купюроприёмника (проверенные и возвращённые купюры не входят в это значение);
- свойство DetectCount возвращает количество купюр, успешно проверенных и возвращённых пользователю в ходе сеанса купюроприёмника;
- метод GetBillCountByDenom() возвращает количество купюр определённого номинала, принятых в ходе сеанса купюроприёмника;
- метод GetBillCounters() возвращает коллекцию счётчиков купюр для каждого принятого номинала (в виде объекта CashMaticItems);
- свойство RejectCount возвращает количество купюр, забракованных и возвращённых пользователю в ходе сеанса купюроприёмника;
- метод ResetCounters() сбрасывает все счётчики купюр сеанса купюроприёмника.

Имя	Тип результата	Тип обращения	Назначение
Accepted	Объект **CashMaticBill**, число	*Свойство, только чтение*	Принятая купюра, номинал. *Возвращаемое значение может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticBill.Value**.* **Значение доступно только при обработке события OnAccept, в другое время обращение к этому свойству вызывает ошибку.** **Совместимость:** поддерживается, начиная с версии **2.6.0**
Active	Логический	*Свойство, чтение/запись*	Управление купюроприёмником: Истинное значение включает приём или проверку купюр (в зависимости от значения свойства **Detect** - см. Режимы проверки купюр и приёма купюр); Ложное значение выключает приём/проверку купюр. **Скрипт обязательно должен явно выключать приём/проверку купюр при завершении сеанса купюроприёмника - в этом случае гарантируется правильная обработка купюры, которая может быть вставлена пользователем в последний момент.** Не изменять во время вызова события **OnActive** (•).

Имя	Тип результата	Тип обращения	Назначение
AddHandler (Handler)	Число	*Метод*	Метод устанавливает (добавляет) обработчик событий сеанса купюроприёмника. Параметр **Handler** - объект, реализующий интерфейс **CashMaticCashSessionEvents**. Возвращаемое численное значение - идентификатор обработчика, используется для последующего удаления обработчика вызовом метода **RemoveHandler**. (см. Примеры обработчиков событий сеанса купюроприёмника) **Совместимость:** поддерживается, начиная с версии **2.6.0**
BillCount	Объект **CashMaticCashCounter**, число	*Свойство, только чтение*	Счётчик принятых купюр в данном сеансе купюроприёмника. *Возвращаемое значение может использоваться как численное, см. свойство **CashMaticCashCounter.Value**.* **Совместимость:** ранее версии **2.6.0** количество купюр возвращается только как число.
BillTotal	Объект **CashMaticCurrency**, число	*Свойство, только чтение*	Количество принятых денег (в денежных единицах) в данном сеансе купюроприёмника. *Возвращаемое значение может использоваться как численное, см. свойство **CashMaticCurrency.Value**.* **Совместимость:** ранее версии **2.6.0** количество денег возвращается только как число.
Cancel()	Пустой	*Метод*	Метод выключает приём/проверку купюр на купюроприёмнике. *Эквивалентно присваиванию ложного значения свойству **Active**.* Не вызывать во время вызова события **OnActive** (•). **Скрипт обязательно должен явно выключать приём/проверку купюр при завершении сеанса купюроприёмника - в этом случае гарантируется правильная обработка купюры, которая может быть вставлена пользователем в последний момент.** **Совместимость:** поддерживается, начиная с версии **2.6.0**
Cash	Объект **CashMaticCash**	*Свойство, только чтение*	Возвращает родительский объект управления купюроприёмником **CashMaticCash**. **Совместимость:** поддерживается, начиная с версии **2.6.0**
Detect	Логическое	*Свойство, чтение/запись*	Управление режимом проверки купюр (см. Режимы проверки купюр и приёма купюр): Истинное значение включает режим проверки купюр (выключая тем самым режим приёма купюр); **В режиме проверки купюр купюра не принимается, а только проверяется - определяется номинал купюры, после чего она возвращается пользователю.** Ложное значение выключает режим проверки купюр (включая тем самым режим приёма купюр). **Включение и выключение режима проверки купюр допустимо только в неактивном состоянии сеанса купюроприёмника (т.е. когда свойство Active имеет ложное значение), иначе изменение свойства Detect вызывает ошибку.** **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
DetectCount	Число	*Свойство, только чтение*	Количество проверенных купюр в данном сеансе купюроприёмника (см. Режимы проверки купюр и приёма купюр). **Проверенные купюры не учитываются счётчиками принятых купюр.** **Совместимость:** поддерживается, начиная с версии **2.6.0**
Detected	Объект CashMaticBill, число	*Свойство, только чтение*	Проверенная купюра, номинал. **Значение доступно только при обработке события OnDetect, в другое время обращение к этому свойству вызывает ошибку.** *Возвращаемое значение может использоваться как численное, равное номиналу купюры, см. свойство CashMaticBill.Value.* **Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).** **Совместимость:** поддерживается, начиная с версии **2.6.0**
GetBillCountByDenom(Denom)	Объект CashMaticBillCounter, число	*Метод*	Возвращает количество принятых купюр указанного номинала в данном сеансе купюроприёмника (параметр **Denom** - численное значение, в денежных единицах). *Возвращаемое значение может использоваться как численное, см. свойство CashMaticBillCounter.Value.* **Совместимость:** ранее версии **2.6.0** количество купюр возвращается только как число.
GetBillCounters()	Объект CashMaticItems	*Метод*	Набор счётчиков купюр, принятых в данном сеансе купюроприёмника, раздельно по типам купюр (отдельный счётчик для каждого типа купюр). Результат возвращается в виде коллекции **CashMaticItems**, элементами коллекции являются объекты **CashMaticBillCounter**. **Совместимость:** поддерживается, начиная с версии **2.6.0**
LastAccepted	Объект CashMaticBill, число	*Свойство, только чтение*	Последняя (на текущий момент) купюра, принятая в данном сеансе купюроприёмника, номинал. *Возвращаемое значение может использоваться как численное, равное номиналу купюры, см. свойство CashMaticBill.Value.* **До приёма первой купюры (до события OnAccept), свойство имеет пустое значение.** *В отличие от свойства Accepted, свойство доступно всегда (в т.ч. и после обработки приёма купюры).* **Совместимость:** поддерживается, начиная с версии **2.6.0**
LastDetected	Объект CashMaticBill, число	*Свойство, только чтение*	Последняя (на текущий момент) купюра, проверенная в данном сеансе купюроприёмника, номинал. *Возвращаемое значение может использоваться как численное, равное номиналу купюры, см. свойство CashMaticBill.Value.* **До проверки первой купюры (до события OnDetect), свойство имеет пустое значение.** *В отличие от свойства Detected, свойство доступно всегда (в т.ч. и после обработки проверки купюры).* **Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).** **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
MaxBill	Объект **CashMaticBill**, число	*Свойство, чтение/запись*	Максимальный допустимый номинал следующей купюры в данном сеансе купюроприёмника. По умолчанию - максимальный поддерживаемый номинал (в текущей версии - `5000` руб.). При изменении значения свойства вызывается событие **OnBillRange**. *Значение может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticBill.Value**.* **Совместимость:** ранее версии **2.6.0** поддерживается только численное значение, событие **OnBillRange** не поддерживается.
MinBill	Объект **CashMaticBill**, число	*Свойство, чтение/запись*	Минимальный допустимый номинал следующей купюры в данном сеансе купюроприёмника. По умолчанию - минимальный поддерживаемый номинал (в текущей версии - `5` руб. *(обычно 5 рублей купюроприёмники не принимают, тем не менее код для такого типа купюр зарезервирован)*). При изменении значения свойства вызывается событие **OnBillRange**. *Значение может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticBill.Value**.* **Совместимость:** поддерживается, начиная с версии **2.6.0**
OnAccept	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при успешном приёме купюры в данном сеансе купюроприёмника. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, принятая купюра доступна через свойство **Accepted**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, Bill)`, где: **CashSession** - сеанс купюроприёмника, в котором произошёл приём купюры (объект **CashMaticCashSession**); **Bill** - принятая купюра, номинал (эквивалентно значению **Accepted**) (объект **CashMaticBill**). **Совместимость:** ранее версии **2.6.0** поддерживается только строковое значение.
OnActive	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при изменении активности данного сеанса купюроприёмника (изменении свойства **Active** или вызове метода **Cancel**), т.е. при включении или выключении приёма/проверки купюр. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, состояние активности доступно через свойство **Active**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, Active)`, где: **CashSession** - сеанс купюроприёмника, в котором произошло изменение активности (объект **CashMaticCashSession**); **Active** - логическое значение, равное новому (изменившемуся) значению свойства **Active**. Не изменять значение свойства **Active** во время вызова события **OnActive** (•). **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
OnBillCount	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при изменении количества принятых купюр в данном сеансе купюроприёмника (вызывается как при успешном приёме купюры, так и при сбросе счётчиков вызовом метода **ResetCounters()**). Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, количество принятых купюр доступно через свойство **BillCount**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, BillCount)`, где: **CashSession** - сеанс купюроприёмника, в котором произошло изменение количества принятых купюр (объект **CashMaticCashSession**); **BillCount** - значение, равное новому (изменившемуся) количеству принятых купюр (объект **CashMaticCashCounter**). *Значение может использоваться как численное, см. свойство **CashMaticCashCounter.Value**.* **Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).** **Совместимость:** поддерживается, начиная с версии **2.6.0**
OnBillRange	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при изменении диапазона допустимых номиналов купюр в данном сеансе купюроприёмника (вызывается как при изменении диапазона через свойство **MinBill**, свойство **MaxBill**, метод **SetBillRange()**, так и при сбросе диапазона вызовом метода **ResetBillRange()**). Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, диапазон допустимых номиналов купюр доступен через свойства **MinBill** и **MaxBill**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, MinBill, MaxBill)`, где: **CashSession** - сеанс купюроприёмника, в котором произошло изменение диапазона допустимых номиналов купюр (объект **CashMaticCashSession**); **MinBill** - значение, равное новому (изменившемуся) минимальному допустимому номиналу купюр (объект **CashMaticBill**), **MaxBill** - значение, равное новому (изменившемуся) максимальному допустимому номиналу купюр (объект **CashMaticBill**). **Совместимость:** поддерживается, начиная с версии **2.6.0**
OnCancel	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при отмене/прекращении приёма/проверки купюр в данном сеансе купюроприёмника. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа); Если задана функция, то при вызове события она вызывается с параметром `(CashSession)`, где: **CashSession** - сеанс купюроприёмника, в котором прекратились приём/проверка купюр. **Совместимость:** ранее версии **2.6.0** поддерживается только строковое значение.

Имя	Тип результата	Тип обращения	Назначение
OnDetect	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при успешной проверке купюры в данном сеансе купюроприёмника. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, принятая купюра доступна через свойство **Accepted**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, Bill)`, где: **CashSession** - сеанс купюроприёмника, в котором произошла проверка купюры (объект **CashMaticCashSession**); **Bill** - проверенная купюра, номинал (эквивалентно значению **Detected**) (объект **CashMaticBill**). **Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).** **Совместимость:** поддерживается, начиная с версии **2.6.0**
OnFailure	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый при ошибке купюроприёмника в данном сеансе. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа); Если задана функция, то при вызове события она вызывается с параметром `(CashSession)`, где: **CashSession** - сеанс купюроприёмника, в котором произошла ошибка. **Совместимость:** ранее версии **2.6.0** поддерживается только строковое значение.
OnReject	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый в случае браковки купюры при приёме/проверке в данном сеансе купюроприёмника (купюра возвращается пользователю). Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа); Если задана функция, то при вызове события она вызывается с параметром `(CashSession)`, где: **CashSession** - сеанс купюроприёмника, в котором произошла браковка купюры. **Совместимость:** ранее версии **2.6.0** поддерживается только строковое значение.
OnTimeout	Строка или функция	*Свойство, чтение/запись*	Скрипт, выполняемый в случае завершения по таймауту приёма/проверки купюры в данном сеансе купюроприёмника. Если задана строка, то при вызове события она выполняется как javascript (в контексте текущего HTML-документа, значение таймаута доступно через свойство **Timeout**); Если задана функция, то при вызове события она вызывается с параметрами `(CashSession, Timeout)`, где: **CashSession** - сеанс купюроприёмника, в котором произошла браковка купюры. **Timeout** - таймаут (превышенное значение, в секундах - соответствует свойству **Timeout**). **Совместимость:** поддерживается, начиная с версии **2.6.0**

Имя	Тип результата	Тип обращения	Назначение
RejectCount	Число	*Свойство, только чтение*	Количество отказов в приёме купюры (см. **OnReject**) в данном сеансе купюроприёмника.
RemoveHandler (Id)	Пустой	*Метод*	Метод снимает (удаляет) обработчик событий сеанса купюроприёмника. Параметр **Id** - идентификатор обработчика (число), полученный при установке обработчика в результате вызова метода **AddHandler**. (см. Примеры обработчиков событий сеанса купюроприёмника) **удаление несуществующего обработчика (неверное значение Id) вызывает ошибку выполнения скрипта.** **Совместимость:** поддерживается, начиная с версии **2.6.0**
ResetBillRange ()	Пустой	*Метод*	Метод сбрасывает ограничения диапазона допустимых номиналов купюр. Минимальный/максимальный допустимые номиналы следующей купюры в данном сеансе купюроприёмника принимают значение по умолчанию - минимальный поддерживаемый номинал (в текущей версии - 5 руб. *(обычно 5 рублей купюроприёмники не принимают, тем не менее код для такого типа купюр зарезервирован)*), максимальный поддерживаемый номинал (в текущей версии - 5000 руб.). При сбросе ограничений допустимых номиналов купюр вызывается событие **OnBillRange**. **Совместимость:** поддерживается, начиная с версии **2.6.0**
ResetCounters ()	Пустой	*Метод*	Метод сбрасывает все счётчики данного сеанса приёма купюр. При сбросе счётчиков принятых купюр вызывается событие **OnBillCount**. *Сбрасываются значения свойств **BillCount**, **BillTotal**, **DetectCount**, **RejectCount**.* *Сбрасываются значения, возвращаемые как результаты вызовов **GetBillCounters()**, **GetBillCountByDenom(Denom)**.* **Совместимость:** поддерживается, начиная с версии **2.6.0**
SetBillRange (MinBill, MaxBill)	Пустой	*Метод*	Метод устанавливает ограничения диапазона допустимых номиналов купюр. При изменении ограничений допустимых номиналов купюр вызывается событие **OnBillRange**. *Вызов **SetBillRange** эквивалентен поочерёдному заданию значений свойств **MinBill**/**MaxBill** (однако при вызове **SetBillRange** событие **OnBillRange** срабатывает однократно).* **Совместимость:** поддерживается, начиная с версии **2.6.0**
Timeout	Число	*Свойство, чтение/запись*	Таймаут активного состояния приёма/проверки купюры, в секундах (по умолчанию - нулевое значение, соответствует отсутствию ограничения). Таймаут отсчитывается от момента присваивания истинного значения свойству **Active**. При превышении таймаута срабатывает событие **OnTimeout**. **Нельзя изменять таймаут при активном состоянии сеанса купюроприёмника (когда свойство Active имеет истинное значение).** **Совместимость:** поддерживается, начиная с версии **2.6.0**

(•) Попытка изменить значение свойства `Active` во время выполнения обработчика события `OnActive` приведёт к программной ошибке с сообщением **"Текущий сеанс купюроприёмника невозможно прервать в данный момент"** или **"Сеанс купюроприёмника невозможно активировать в данный момент"**.

Совместимость: ошибка вбрасывается, начиная с версии **2.7.2**, в предыдущих версиях это действие всё равно запрещено, может быть проигнорировано или выполнено неправильно.

7.2.7 Объект CashMaticCurrency, свойства

см. также:	CashMaticBill	CashMaticCash
	CashMaticCashCounter	CashMaticCashSession

Объект CashMaticCurrency описывает денежное значение с включением кода валюты.
Код валюты (свойство Code) указывается буквенной комбинацией, соответствующей стандарту **ISO 4217**.
В текущей версии поддерживается только российский рубль (**"RUB"**).
Денежное значение (в единицах соответствующей валюты) доступно как свойство Value.
Наличие свойства Value позволяет использовать объект в скриптах как значение свойства Value.

В текущей версии объекты CashMaticCurrency создаются как результат следующих вызовов:

- метод BillTypeToMoney() объекта CashMaticCash;
- свойство DefaultCurrency объекта CashMaticCash;
- свойство Total объекта CashMaticCashCounter;
- свойство BillTotal объекта CashMaticCashSession.

В текущей версии свойства объекта CashMaticCurrency доступны только для чтения, в дальнейшем предполагается предоставить возможность разработчику прикладного скрипта изменять значения свойств.

Родственными по смыслу объектами являются:

- CashMaticBill - денежный номинал одной купюры;
- CashMaticBillCounter - суммарное значение некоторого количества купюр одного достоинства;
- CashMaticCashCounter - сумма купюр произвольного (одного или разного) достоинства.

Имя	Тип результата	Тип обращения	Назначение
Code	Строка	*Свойство, только чтение*	код валюты по стандарту ISO 4217.
Value	Число	*Свойство, только чтение*	денежное значение. *(наличие поля **Value** позволяет в скриптах использовать объект **CashMaticCurrency** как число)*

Совместимость: объект CashMaticCurrency поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.8 Объект CashMaticItems, свойства и методы

см. также: CashMaticCashSession CashMaticSaleItems

Объект CashMaticItems представляет собой коллекцию, содержащую известное неизменяемое число некоторых элементов.
Объект имеет как методы, типичные как для COM-коллекции (Item(), Count()), так и для свойство, типичное для javascript-коллекции: length.
Прикладной скрипт или компонент расширения не могут создать произвольную коллекцию CashMaticItems, а также не могут изменять состав или порядок элементов в существующей коллекции.

В текущей версии **NDemia CashMatic** объект CashMaticItems используется для представления результата вызова метода GetBillCounters() объекта CashMaticCashSession, поэтому элементы коллекции всегда определённо имеют тип CashMaticBillCounter.
В дальнейшем предполагается расширение применения объекта CashMaticItems, тип элементов должен определяться из контекста использования.

Имя	Тип результата	Тип обращения	Назначение
Count()	Число	*Метод*	Число элементов в коллекции (значение аналогично свойству **length)**
Item(index)	Объект	*Метод*	Возвращает элемент коллекции под номером **index** (начиная с нулевого)
length	Число	*Свойство, только чтение*	Число элементов в коллекции (значение аналогично методу **Count**)

Совместимость: объект CashMaticItems поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.9 Объект CashMaticRequest, свойства и методы

см. также: CashMaticCash CashMaticResult CashMaticResultHandler

Объект CashMaticRequest является средством управления выполнением асинхронного запроса (в текущей версии асинхронные запросы создаются методами создаются методами GetDeviceProtocol() и GetDeviceIdentification() объекта CashMaticCash, в дальнейшем предполагается расширение применения этого механизма в **NDemia CashMatic**).

Объект создаётся в результате запуска асинхронного запроса (см. Примеры опроса идентификации купюроприёмника).

Удаление объекта приводит к отмене запроса. Однако используемый в javascript механизм "сборки мусора" (garbage collection) может при освобождении всех ссылок на объект задерживать удаление объекта на неопределённое время, поэтому для отмены (досрочного завершения) запроса предусмотрен метод CancelRequest().
Свойство RequestCompleted является логическим флагом (признаком) завершения запроса, значение меняется с ложного на истинное при завершении запроса (с любым результатом, по любой причине).
Свойство ResultHandler содержит ссылку на указанный при запуске асинхронного запроса обработчик результата запроса (интерфейс CashMaticResultHandler). После запуска запроса обработчик не может быть удалён или заменён до завершения запроса. При завершении запроса результат выполнения запроса передаётся этому обработчику.

Имя	Тип результата	Тип обращения	Назначение
CancelRequest()	Пустой	*Метод*	Прервать или отменить выполнение запроса. Выполняется нормальная обработка завершения запроса: обработчик **ResultHandler** вызывается (с результатом, соответствующим отмене - в зависимости от спецификации запроса); свойство **RequestCompleted** принимает истинное значение.
RequestCompleted	Логический	*Свойство, только чтение*	Флаг завершения запроса. ложное значение: запрос не завершён (выполняется); истинное значение: запрос завершён (выполнен).
RequestId	Число	*Свойство, только чтение*	Идентификатор запроса. Каждый запрос (объект **CashMaticRequest**) имеет постоянный во времени ненулевой идентификатор, уникальный среди одновременно существующих запросов. *В текущей версии - только внутреннее использование.*
RequestSubject	Строка	*Свойство, только чтение*	Кратко сформулированный предмет запроса (в человекочитаемом виде, для отладочных целей).
ResultHandler	Объект **CashMaticResultHandler**	*Свойство, только чтение*	Обработчик результата запроса - объект **CashMaticResultHandler**.

Совместимость: объект CashMaticRequest поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.2.10 Объект CashMaticResult, свойства

см. также: CashMaticCash CashMaticRequest CashMaticResultHandler

Объект CashMaticResult используется для передачи результата выполнения асинхронного запроса обработчику результата (в текущей версии асинхронные запросы создаются методами создаются методами GetDeviceProtocol() и GetDeviceIdentification() объекта CashMaticCash, в дальнейшем предполагается расширение применения этого механизма в **NDemia CashMatic**).

Прикладной скрипт или компонент расширения не могут создать объект CashMaticResult, но могут получить его через интерфейс обработчика результата запроса CashMaticResultHandler (см. Примеры опроса идентификации купюроприёмника).
Все свойства объекта CashMaticResult доступны только на чтение, поскольку описывают результат уже завершившегося запроса, изменение свойств не имеет прикладного смысла.

Основным является свойство Data, конкретный его тип зависит от вида выполненного запроса (тип результата должен описываться в спецификации запроса).
Обработчик результата запроса должен проверять код результата (свойство Code), нулевой код соответствует нормальному завершению, другие значения указывают на ошибки, набор возможных значений определяется видом запроса.
Для упрощения формирования текстовых сообщений по результатам выполнения запроса используется свойство Description (разъяснение результата запроса в человекочитаемом текстовом виде).

Имя	Тип результата	Тип обращения	Назначение
Code	Число	*Свойство, только чтение*	Код завершения запроса (код результата). Набор возможных значений определяется видом запроса. Нулевой код должен соответствовать положительному результату.
Data	Объект	*Свойство, только чтение*	Данные результата запроса. Тип и значение определяются видом запроса. Обращение к свойству допустимо только после завершения запроса с положительным результатом.
Description	Строка	*Свойство, только чтение*	Текстовое разъяснение результата запроса (человекочитаемый текст).
Result	Строка	*Свойство, только чтение*	Результат запроса, необработанное внутреннее представление. Свойство **Result** имеет только отладочное назначение, для прикладных целей должны использоваться свойства **Code** и **Data**.
Subject	Строка	*Свойство, только чтение*	Кратко сформулированный предмет запроса (в человекочитаемом виде, для отладочных целей).

Совместимость: объект CashMaticResult поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.3 Интерфейсы

CashMaticCashSessionEvents
CashMaticResultHandler

7.3.1 Интерфейс CashMaticCashSessionEvents, методы

см. также: CashMaticCash CashMaticCashSession

Интерфейс CashMaticCashSessionEvents не реализуется собственными объектами **NDemia CashMatic**.
Этот интерфейс должен реализовываться объектом или объектами прикладных скриптов, разрабатываемых на основе **NDemia CashMatic**.
Объект, реализующий интерфейс CashMaticCashSessionEvents, называется обработчиком событий сеанса купюроприёмника (см. Примеры обработчиков событий сеанса купюроприёмника).
Сеанс купюроприёмника (объект CashMaticCashSession) имеет методы установки (добавления) обработчиков (AddHandler) и удаления обработчиков (RemoveHandler).
Установка обработчиков событий сеанса купюроприёмника не является обязательной.
Может быть установлено один и более одного обработчиков событий от одного источника (сеанса купюроприёмника).
Каждый обработчик событий, установленный методом AddHandler объекта CashMaticCashSession, должен быть удалён методом RemoveHandler объекта CashMaticCashSession.
Допускается многократная установка одного обработчика событий, число удалений должно соответствовать числу установок.
Обработчик событий уведомляется о событиях сеанса купюроприёмника посредством вызовов соответствующих методов.
Установка обработчика событий методом AddHandler и удаление обработчика событий методом RemoveHandler допускаются в любое время, в том числе во время выполнения метода данного или другого обработчика событий.
Во время выполнения методов обработчик событий не должен без существенных причин изменять объект-источник событий (сеанс купюроприёмника, CashMaticCashSession), поскольку это может породить вложенные события с вызовом других методов (таким образом, будет нарушена естественная последовательность событий, что создаёт трудности для разработки и отладки стабильно работающего приложения).
Обработчик событий не должен полагаться на отсутствие других обработчиков в том же сеансе купюроприёмника (они могут быть, например, неявно установлены компонентами расширений).

Если одновременно установлено несколько (более одного) обработчиков событий:
- не требуется соблюдение какого-либо соответствия между порядком добавления и порядком удаления обработчиков событий (кроме точного баланса вызовов AddHandler/RemoveHandler);
- все установленные обработчики событий будут в равной мере уведомляться о событиях сеанса купюроприёмника;
- не гарантируется какой-либо определённый порядок уведомления различных обработчиков;
- никакой из установленных обработчиков событий не может препятствовать уведомлению всех остальных установленных обработчиков событий или влиять на это уведомление каким-либо образом.

Имя	Тип результата	Тип обращения	Назначение
OnAccept(CashSession, Bill)	Пустой	*Метод*	Метод вызывается при успешном приёме купюры **Bill** (номинал купюры, объект **CashMaticBill**) в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**). *Значение **Bill** может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticBill.Value**.*
OnActive(CashSession, Active)	Пустой	*Метод*	Метод вызывается при изменении активности сеанса купюроприёмника (изменении свойства **Active** или вызове метода **Cancel**), т.е. при включении или выключении приёма/проверки купюр. Параметр **CashSession** - сеанс купюроприёмника (объект **CashMaticCashSession**); Параметр **Active** - логическое значение, равное новому (изменившемуся) значению свойства **CashMaticCashSession.Active**.
OnBillCount(CashSession, BillCount)	Пустой	*Метод*	Метод вызывается при изменении количества принятых купюр в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**), как при успешном приёме купюры, так и при сбросе счётчиков вызовом **CashMaticCashSession.ResetCounters()**. Параметр **BillCount** - значение, равное новому (изменившемуся) количеству принятых купюр (объект **CashMaticCashCounter**). *Значение **BillCount** может использоваться как численное, см. свойство **CashMaticCashCounter.Value**.* **Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).**
OnBillRange(CashSession, MinBill, MaxBill)	Пустой	*Метод*	Метод вызывается при изменении диапазона допустимых номиналов купюр в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**), как при изменении диапазона через свойство **CashMaticCashSession.MinBill**, свойство **CashMaticCashSession.MaxBill**, метод **CashMaticCashSession.SetBillRange(MinBill, MaxBill)**, так и при сбросе диапазона вызовом метода **CashMaticCashSession.ResetBillRange()**. Параметр **CashSession** - сеанс купюроприёмника, в котором произошло изменение диапазона допустимых номиналов купюр (объект **CashMaticCashSession**); Параметр **MinBill** - значение, равное новому (изменившемуся) минимальному допустимому номиналу купюр (объект **CashMaticBill**); Параметр **MaxBill** - значение, равное новому (изменившемуся) максимальному допустимому номиналу купюр (объект **CashMaticBill**). *Значения **MinBill** и **MaxBill** могут использоваться как численные (номиналы купюр), см. свойство **CashMaticBill.Value**.*
OnCancel(CashSession)	Пустой	*Метод*	Метод вызывается при отмене/прекращении приёма/проверки купюр в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**).
OnDetect(CashSession, Bill)	Пустой	*Метод*	Метод вызывается при успешной проверке купюры **Bill** (номинал купюры, объект **CashMaticBill**) в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**). *Значение **Bill** может использоваться как численное, равное номиналу купюры, см. свойство **CashMaticBill.Value**.*

			Проверенные купюры не учитываются счётчиками принятых купюр (см. Режимы проверки купюр и приёма купюр).
OnFailure(CashSession)	Пустой	*Метод*	Метод вызывается при ошибке купюроприёмника в сеансе **CashSession** (объект **CashMaticCashSession**).
OnReject(CashSession)	Пустой	*Метод*	Метод вызывается в случае браковки купюры при приёме/проверке в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**). Бракованная купюра возвращается пользователю. Номинал купюры недоступен (либо купюроприёмник его не определил, либо определил как выходящий за пределы диапазона допустимых номиналов - см. объект **CashMaticCashSession**, свойства **MinBill** и **MaxBill**, методы **SetBillRange** и **ResetBillRange**).
OnTimeout(CashSession, Timeout)	Пустой	*Метод*	Метод вызывается в случае завершения по таймауту приёма/проверки купюры в сеансе купюроприёмника **CashSession** (объект **CashMaticCashSession**). Параметр **Timeout** - таймаут (превышенное значение, в секундах).

Совместимость: интерфейс `CashMaticCashSessionEvents` поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.3.2 Интерфейс CashMaticResultHandler, методы

см. также: CashMaticCash CashMaticRequest CashMaticResult

Интерфейс `CashMaticResultHandler` служит для организации асинхронных запросов (в текущей версии асинхронные запросы создаются методами создаются методами `GetDeviceProtocol()` и `GetDeviceIdentification()` объекта `CashMaticCash`, в дальнейшем предполагается расширение применения этого механизма в **NDemia CashMatic**).

`CashMaticResultHandler` не является собственным объектом **NDemia CashMatic**, этот интерфейс должен реализовываться объектом или объектами прикладных скриптов, разрабатываемых на основе **NDemia CashMatic**.
Объект `CashMaticResultHandler` называется обработчиком результата запроса (см. Примеры опроса идентификации купюроприёмника).

Метод `HandleResult(Result)` вызывается при завершении асинхронного запроса (в том числе при отмене или отрицательном результате), результат запроса передаётся в параметре `Result` как объект `CashMaticResult` (см. описание свойств объекта CashMaticResult).

Имя	Тип результата	Тип обращения	Назначение
HandleResult(Result)	Пустой	*Метод*	Обработка результата запроса. Параметр **Result** - объект **CashMaticResult**, результат запроса доступен через свойства этого объекта.

Совместимость: интерфейс `CashMaticResultHandler` поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

7.4 Примеры

7.4.1 Примеры обработчиков событий сеанса купюроприёмника

Сеанс купюроприёмника - это процесс, в ходе которого купюроприёмник принимает купюры от пользователя. С точки зрения общего сценария платежа (сценария работы пользователя) приём купюр часто удобно рассматривать как одно целостное событие или состояние: "пользователь вносит деньги". На самом деле за время этого состояния происходит ряд переключений - сеанс купюроприёмника состоит из одного и более циклов включения-выключения приёма купюр на купюроприёмнике, причём каждый цикл может отличаться от других диапазоном принимаемых купюр, таймаутом и т.п.
NDemia CashMatic сообщает прикладному скрипту о различных событиях, связанных с купюроприёмником и соответствующих различным изменениям параметров, успешному или неуспешному приёму купюр и т.п.
Для получения этих событий прикладная программа (скрипт) должна создать собственные обработчики. Исходя из основного назначения купюроприёмника (приём и проверка купюр), самыми главными являются события приёма купюры (`OnAccept`) или проверки купюры (`OnDetect`) объекта `CashMaticCashSession`.
Следует обратить особое внимание на завершение приёма или проверки купюр, поскольку при некорректном завершении можно потерять информацию о последней купюре, введённой "почти одновременно" с моментом завершения.
Во избежание потери последней купюры прикладная программа должна обязательно завершать режим приёма/проверки купюр явным вызовом

```
CashSession.Active = false;
```

или

```
CashSession.Cancel();
```

где `CashSession` - это объект `CashMaticCashSession`, созданный как `CashMatic.Cash.CreateSession()`;
В этом случае купюры больше не будут приниматься купюроприёмником, несмотря на существование объекта `CashMaticCashSession` (используемый в javascript механизм "сборки мусора" (garbage collection) может задерживать удаление объекта, даже если все ссылки на объект освобождены).

Далее приведены примеры реализации обработчиков событий в прикладном скрипте, для следующих случаев:

1. Строка javascript как обработчик события сеанса купюроприёмника;
2. Функция javascript как обработчик события сеанса купюроприёмника;
3. Объектный обработчик событий сеанса купюроприёмника.

7.4.1.1 Строка javascript как обработчик события сеанса купюроприёмника

В данном примере обработчиком события является строка, содержащая исходный код javascript, который должен выполняться при вызове соответствующего события.
Код javascript выполняется в контексте текущего HTML-документа на момент выполнения.
(см. также Примеры обработчиков событий сеанса купюроприёмника)

```
var CashMatic = external;
        //CashMatic - рекомендуемое имя для основного объекта
        //объектной модели NDemia CashMatic

var CashSession = CashMatic.Cash.CreateSession();
        //создаётся сеанс купюроприёмника, объект CashMaticCashSession

function Пример_инициализация()
        //      эта функция должна быть вызвана
        //для включения приёма купюр на купюроприёмнике.
{
        CashSession.OnAccept = "alert('Получена купюра '
                + CashSession.Accepted + ' рублей');";
                        //этот код будет выполнен при приёме купюры

        CashSession.Timeout = 15;
                //таймаут приёма купюры задаётся в секундах,
                //      по умолчанию равен нулю
                //      (задавать значение не обязательно),
                //      нулевой таймаут воспринимается программой как
                //      отсутствие ограничения по времени для приёма купюры.

        CashSession.OnTimeout = "alert('Купюра не получена за '
                + CashSession.Timeout + ' с')";
                        //этот код будет выполнен, если купюра не будет принята
                        //      за 15 секунд.

        CashSession.Detect = false;
                //выключаем режим проверки купюр
                //      (это действие не является обязательным,
                //      режим проверки выключен по умолчанию)

        //Включаем приём купюр на купюроприёмнике:
        CashSession.Active = true;
}

function Пример_завершение()
        //      эта функция должна быть вызвана
        //для выключения приёма купюр на купюроприёмнике.
{
        //Выключаем приём купюр на купюроприёмнике:
        CashSession.Cancel();

        //Прикладной скрипт должен обязательно выключать приём купюр явно,
        //      не полагаясь на выключение при удалении объекта CashSession,
        //      иначе возможны потери событий,
        //              в том числе потери принятых купюр.

        alert("Приём купюр завершён, всего принято купюр: "
                + CashSession.BillCount + " шт. на сумму "
                + CashSession.BillTotal + " руб.");
}
```

7.4.1.2 Функция javascript как обработчик события сеанса купюроприёмника

В данном примере обработчиком события является функция javascript, которая должна выполняться при вызове соответствующего события.
Функция выполняется в контексте содержащего её HTML-документа.
(см. также Примеры обработчиков событий сеанса купюроприёмника)

```
var CashMatic = external;
	//CashMatic - рекомендуемое имя для основного объекта
	//объектной модели NDemia CashMatic

var CashSession = CashMatic.Cash.CreateSession();
	//создаётся сеанс купюроприёмника, объект CashMaticCashSession

function Пример_инициализация()
	//	эта функция должна быть вызвана
	//для включения приёма купюр на купюроприёмнике.
{
	CashSession.OnAccept = function(CashSession, Bill)
		//эта функция выполняется при приёме каждой купюры.
		{
			alert("Получена купюра " + Bill + " рублей");
		};

	CashSession.Timeout = 15;
		//таймаут приёма купюры задаётся в секундах,
		//	по умолчанию равен нулю
		//	(задавать значение не обязательно),
		//	нулевой таймаут воспринимается программой как
		//	отсутствие ограничения по времени для приёма купюры.

	CashSession.OnTimeout = function(CashSession, Timeout)
		//эта функция выполняется в случае, если купюра не была принята
		//	за отведённое для этого время.
		{
			alert("Купюра не получена за " + Timeout + " с");
		};

	CashSession.Detect = false;
		//выключаем режим проверки купюр
		//	(это действие не является обязательным,
		//	режим проверки выключен по умолчанию)

	//Включаем приём купюр на купюроприёмнике:
	CashSession.Active = true;
}

function Пример_завершение()
	//	эта функция должна быть вызвана
	//для выключения приёма купюр на купюроприёмнике.
{
	//Выключаем приём купюр на купюроприёмнике:
	CashSession.Cancel();

	//Прикладной скрипт должен обязательно выключать приём купюр явно,
	//	не полагаясь на выключение при удалении объекта CashSession,
	//	иначе возможны потери событий,
	//		в том числе потери принятых купюр.

	alert("Приём купюр завершён, всего принято купюр: "
		+ CashSession.BillCount + " шт. на сумму "
		+ CashSession.BillTotal + " руб.");
}
```

7.4.1.3 Объектный обработчик событий сеанса купюроприёмника

В данном примере обработчиком события является объект, реализующий интерфейс `CashMaticCashSessionEvents`.
Реализация обработчика представляется несколько более сложной, чем в предыдущих примерах, однако этот механизм является намного более гибким.
(см. также Примеры обработчиков событий сеанса купюроприёмника)

```
var CashMatic = external;
        //CashMatic - рекомендуемое имя для основного объекта
        //объектной модели NDemia CashMatic

var CashSession = CashMatic.Cash.CreateSession();
        //создаётся сеанс купюроприёмника, объект CashMaticCashSession

var EventHandlerId = 0;                 //идентификатор обработчика событий,
        //нулевое значение показывает отсутствие установленного обработчика

function Пример_инициализация()
        //      эта функция должна быть вызвана
        //для включения приёма купюр на купюроприёмнике.
{
        EventHandlerId = CashSession.AddHandler(
                new EventHandler("пример обработчика"));
                //создаётся и устанавливается наш
                //      объект-обработчик событий (EventHandler)
                //идентификатор установленного обработчика
                //      запоминается как EventHandlerId

        CashSession.Timeout = 15;
                //таймаут приёма купюры задаётся в секундах, по умолчанию
                //      равен нулю (задавать значение не обязательно),
                //      нулевой таймаут воспринимается программой как
                //      отсутствие ограничения по времени для приёма купюры.

        CashSession.Detect = false;
                //выключаем режим проверки купюр (это действие не является
                //      обязательным, режим проверки выключен по умолчанию)

        //Включаем приём купюр на купюроприёмнике:
        CashSession.Active = true;
}

function Пример_завершение()
        //      эта функция должна быть вызвана
        //для выключения приёма купюр на купюроприёмнике.
{
        //Выключаем приём купюр на купюроприёмнике:
        CashSession.Cancel();

        //Прикладной скрипт должен обязательно выключать приём купюр явно,
        //      не полагаясь на выключение при удалении объекта CashSession,
        //      иначе возможны потери событий,
        //              в том числе потери принятых купюр.

        if(EventHandlerId)      //проверяем, был ли установлен наш обработчик
        {
                //удаляется ранее установленный обработчик событий:
                CashSession.RemoveHandler(EventHandlerId);
                EventHandlerId = 0;
        }

        alert("Приём купюр завершён, всего принято купюр: "
                + CashSession.BillCount + " шт. на сумму "
                + CashSession.BillTotal + " руб.");
}

function EventHandler(title)
        /*
```

```
		Эта функция инициализирует объект-обработчик событий,
			реализующий интерфейс CashMaticCashSessionEvents.

		Параметр title показывает, как прикладной скрипт может
			добавить к создаваемому объекту собственные
			свойства, не предусмотренные спецификацией
			интерфейса CashMaticCashSessionEvents.
	*/
{
	this.OnAccept = function(CashSession, Bill)
		//эта функция выполняется при приёме каждой купюры.
		{
			this.ShowResult("Получена купюра " + Bill + " рублей");
		};

	this.OnTimeout = function(CashSession, Timeout)
		//эта функция выполняется в случае, если купюра не была принята
		//	за отведённое для этого время.
		{
			this.ShowResult("Купюра не получена за " + Timeout + " с");
		};

	this.ShowResult = function(text)
		//Дополнительная функция (метод) объекта EventHandler,
		//	не предусмотренная спецификацией интерфейса
		//	CashMaticCashSessionEvents.
		{
			alert(this.title + ": " + text);
		};

	this.title = title;
		//Дополнительное свойство объекта EventHandler,
		//	не предусмотренное спецификацией интерфейса
		//	CashMaticCashSessionEvents.

	//допускается частичная реализация интерфейса
	//	CashMaticCashSessionEvents, поэтому мы можем не назначать
	//	код для обработки остальных методов интерфейса (OnActive,
	//	OnBillCount, OnBillRange, OnCancel, OnDetect, OnFailure,
	//	OnReject)
}
```

7.4.2 Примеры опроса идентификации купюроприёмника

Опрос идентификации купюроприёмника может выполняться прикладным скриптом в целях получения версии прошивки, серийного номера и уникального кода купюроприёмника (см. объект `CashMaticCashDeviceIdentification`)
Эти данные доступны только для купюроприёмника, работающего по протоколу управления CCNET (протокол управления ICT004 не предусматривает идентификации оборудования).
Таким образом, для получения идентификации необходимо сначала определить протокол.

Как протокол, так и идентификация оборудования могут быть получены двумя способами:

1. синхронный (обычный вызов функции с ожиданием готовности результата и непосредственным получением результата в точке вызова);
2. асинхронный (вызов функции, только регистрирующей объект-обработчик результата запроса).

Асинхронный способ существенно сложнее в использовании, однако при этом уменьшается задержка, блокирующая пользовательский интерфейс на время получения данных.
Вообще говоря, заметные задержки при опросе протокола и идентификации купюроприёмника возникают главным образом только в случае неправильной работы аппаратного или программного обеспечения (зависание службы **NDemia CashMatic Kiosk**, отключение купюроприёмника и т.п.). Однако опрос протокола и идентификации выполняется обычно либо при тестировании оборудования, либо при инициализации программы, т.е. когда нет уверенности в том, что всё работает правильно.
Таким образом, для простоты на ранней стадии разработки рекомендуется использовать синхронный способ. Для уменьшения влияния внешних (по отношению к прикладной программе) сбоев рекомендуется использовать более сложный асинхронный способ.

Далее приведены примеры javascript-кода для следующих задач:

1. Непосредственный (синхронный) опрос протокола управления купюроприёмника;
2. Опрос протокола управления купюроприёмника с отложенной обработкой результата;
3. Непосредственный (синхронный) опрос серийного номера купюроприёмника;
4. Опрос серийного номера купюроприёмника с отложенной обработкой результата.

7.4.2.1 Непосредственный (синхронный) опрос протокола управления купюроприёмника

На время получения результата (требуется взаимодействие со службой **NDemia CashMatic Kiosk**) пользовательский интерфейс при синхронном опросе блокируется.
Асинхронный (неблокирующий) опрос с отложенной обработкой результата - см. следующий пример.

```
var CashMatic = external;
        //CashMatic - рекомендуемое имя для основного объекта
        //объектной модели NDemia CashMatic

function Пример()
{
        alert(CashMatic.Cash.GetDeviceProtocol());
                //в результате при выполнении функции Пример будет выдано
                //      сообщение "CCNET" или "ICT004", в зависимости от
                //      настроенных параметров подключения купюроприёмника.

}
```

7.4.2.2 Опрос протокола управления купюроприёмника с отложенной обработкой результата

Прикладной скрипт после вызова функции Пример() может выполнять другие задачи или вернуть управление пользовательскому интерфейсу.
Обработчик результата запроса будет вызван автоматически при готовности результата или при отмене/ошибке выполнения запроса.

```
var CashMatic = external;
	//CashMatic - рекомендуемое имя для основного объекта
	//объектной модели NDemia CashMatic

var Request = null;	//ссылка на асинхронный запрос
			//	(объект CashMaticRequest)

function Пример()
{
	Request = CashMatic.Cash.GetDeviceProtocol(
		new ResultHandler("пример обработчика")
	);
	/*
		метод GetDeviceProtocol при вызове с параметром:
			1. интерпретирует параметр как ссылку на обработчик
				результата запроса;
			2. создаёт асинхронный запрос;
			3. возвращает ссылку на асинхронный запрос,
				не дожидаясь результата запроса.

		В качестве обработчика результата мы создаём
			объект ResultHandler (см. ниже).
		Полученная ссылка на запрос сохраняется
			в переменной Request.
	*/
}

function ResultHandler(title)
	/*
		Эта функция инициализирует объект-обработчик запроса,
			реализующий интерфейс CashMaticResultHandler.

		Параметр title показывает, как прикладной скрипт может
			добавить к создаваемому объекту собственные
			свойства, не предусмотренные спецификацией
			интерфейса CashMaticResultHandler.
	*/
{
	this.HandleResult = function(Result)
		//Функция обработки результата запроса, реализует
		//	соответствующий метод интерфейса CashMaticResultHandler.
		//Эта функция будет автоматически вызвана при готовности
		//	результата нашего запроса (в том числе при
		//	отрицательном результате, т.е. при ошибке или
		//	отмене запроса).
		//Результат запроса передаётся в параметре Result как
		//	объект CashMaticResult.
		{
			if(Result.Code == 0)
			{
				//нулевой код результата означает отсутствие ошибок:
				this.ShowResult(Result.Data);
					//вызываем демонстрационный метод нашего обработчика
					//	для отображения полученного результата.
					//Result.Data содержит строку "CCNET" или "ICT004",
					//	в зависимости от настроенных параметров
					//	подключения купюроприёмника.
			}
			else
				this.ShowResult(
					"Ошибка опроса протокола управления купюроприёмника"
```

```
                                );
                };

        this.ShowResult = function(text)
                //Дополнительная функция нашего объекта,
                //      не предусмотренная спецификацией CashMaticResultHandler.
                {
                        alert(this.title + ": " + text);
                };

        this.title = title;
                //Дополнительное свойство нашего объекта,
                //      не предусмотренное спецификацией CashMaticResultHandler.
}
```

7.4.2.3 Непосредственный (синхронный) опрос серийного номера купюроприёмника

На время получения результата (требуется взаимодействие с купюроприёмником) пользовательский интерфейс при синхронном опросе блокируется.
Асинхронный (неблокирующий) опрос с отложенной обработкой результата - см. следующий пример.

> *Если во время опроса идентификации устройства выполняется активный сеанс приёма или проверки купюры, то идентификация купюроприёмника будет получена по окончании сеанса приёма или проверки купюры. Ожидание данных ограничено таймаутом (по умолчанию 5 секунд) — если за это время активный сеанс не завершится, то опрос идентификации закончится неудачей (будет получен ненулевой код ошибки). При синхронном запросе пользовательский интерфейс блокируется на всё время ожидания.*

```
var CashMatic = external;
        //CashMatic - рекомендуемое имя для основного объекта
        //объектной модели NDemia CashMatic

function Пример()
{
        var SerialNumber;

        //опрашиваем протокол управления купюроприёмника, идентификация
        //      работает только для CCNET.
        if(CashMatic.Cash.GetDeviceProtocol() == "CCNET")
                SerialNumber = CashMatic.Cash.GetDeviceIdentification().
                        SerialNumber;
                //метод GetDeviceIdentification() возвращает объект
                //      CashMaticCashDeviceIdentification


        if(SerialNumber)
                alert("Серийный номер купюроприёмника: " + SerialNumber);
        else
                alert("Не удалось определить номер купюроприёмника");
}
```

7.4.2.4 Опрос серийного номера купюроприёмника с отложенной обработкой результата

Прикладной скрипт после вызова функции Пример() может выполнять другие задачи или вернуть управление пользовательскому интерфейсу.
Обработчик результата запроса будет вызван автоматически при готовности результата или при отмене/ошибке выполнения запроса.

> *Если во время опроса идентификации устройства выполняется активный сеанс приёма или проверки купюры, то идентификация купюроприёмника будет получена по окончании сеанса приёма или проверки купюры. Ожидание данных ограничено таймаутом (по умолчанию 5 секунд) — если за это время активный сеанс не завершится, то опрос идентификации закончится неудачей (будет получен ненулевой код ошибки). При асинхронном запросе пользовательский интерфейс на это время не блокируется.*

```
var CashMatic = external;
	//CashMatic - рекомендуемое имя для основного объекта
	//объектной модели NDemia CashMatic

var Request = null;	//ссылка на асинхронный запрос
			//	(объект CashMaticRequest)

function Пример()
{
	Request = CashMatic.Cash.GetDeviceIdentification(
		new ResultHandler("пример обработчика")
	);
	/*
		Для упрощения примера пропускаем опрос протокола управления
			купюроприёмника
		(при необходимости см. предыдущие примеры)

		метод GetDeviceIdentification при вызове с параметром:
			1. интерпретирует параметр как ссылку на обработчик
				результата запроса;
			2. создаёт асинхронный запрос;
			3. возвращает ссылку на асинхронный запрос,
				не дожидаясь результата запроса.

		В качестве обработчика результата мы создаём
			объект ResultHandler (см. ниже);
		Полученная ссылка на запрос сохраняется
			в переменной Request.
	*/
}

function ResultHandler(title)
	/*
		Эта функция инициализирует объект-обработчик запроса,
			реализующий интерфейс CashMaticResultHandler.

		Параметр title показывает, как прикладной скрипт может
			добавить к создаваемому объекту собственные
			свойства, не предусмотренные спецификацией
			интерфейса CashMaticResultHandler.
	*/
{
	this.HandleResult = function(Result)
		//Функция обработки результата запроса, реализует
		//	соответствующий метод интерфейса CashMaticResultHandler.
		//Эта функция будет автоматически вызвана при готовности
		//	результата нашего запроса (в том числе при
		//	отрицательном результате, т.е. при ошибке или
		//	отмене запроса).
		//Результат запроса передаётся в параметре Result как
		//	объект CashMaticResult.
```

```
		{
			//нулевой код результата означает отсутствие ошибок:
			if(Result.Code == 0)
			{
				this.ShowResult(Result.Data.SerialNumber);
					//вызываем демонстрационный метод нашего обработчика
					//	для отображения полученного результата.
					//Result.Data содержит объект
					//	CashMaticCashDeviceIdentification,
					//	серийный номер купюроприёмника доступен
					//	через свойство SerialNumber этого объекта.
			}
			else
				this.ShowResult(
					"Ошибка опроса идентификации купюроприёмника"
				);
		};

	this.ShowResult = function(text)
		//Дополнительная функция нашего объекта,
		//	не предусмотренная спецификацией CashMaticResultHandler.
		{
			alert(this.title + ": " + text);
		};

	this.title = title;
		//Дополнительное свойство нашего объекта,
		//	не предусмотренное спецификацией CashMaticResultHandler.
}
```

8 Сеанс приёма платежа

см. также:	Компоненты расширения	Прикладные скрипты	Приём наличных денег
	Модель навигации	Модель обработки отказов	Переменные платёжного сеанса при печати
	Интерфейс ICashMaticSessionExtender	Объект CashMaticSession	Пример платёжного интерфейса

Обычно в любой платёжной системе можно выделить некоторый интервал времени, в течение которого один конкретный пользователь работает с терминалом - сообщает и получает какие-либо сведения, вносит наличные деньги, получает печатные документы. Какими бы ни были назначение и сложность платёжной системы, всё равно - всё сводится к существованию этого интервала, называемого сеансом приёма платежа, или платёжным сеансом.
За платёжный сеанс отвечает объект CashMaticSession.
Объект CashMaticSession существует в контексте работы хост-приложения, а не HTML-документа, скрипт может лишь получить ссылку на этот объект, но не может его создать или уничтожить. Используя методы AddProp/GetProp, скрипты могут записывать и читать именованные значения (переменные сеанса), которые будут сохраняться при переходах со страницы на страницу (кроме того, эти переменные доступны для шаблонов печати выходных документов). Компоненты расширения через эти методы могут взаимодействовать со скриптом, изменяя контекст платёжного сеанса.

Вызов метода Reset должен соответствовать времени начала ожидания работы интерактивного пользователя (плательщика).
Вызов метода Start должен соответствовать времени начала работы интерактивного пользователя (плательщика).
Между вызовами Reset и Start может пройти сколь угодно длительное время (зависит от наличия и активности пользователей платёжного терминала).
Создание переменных платёжного сеанса можно начинать после вызова метода Reset, однако переменные, связанные с логикой действий интерактивного пользователя (плательщика), следует создавать после вызова Start.

Метод Reset полностью сбрасывает контекст платёжного сеанса - все созданные переменные удаляются. Рекомендуется вызывать метод Reset явно - в текущей версии это не обязательно (автоматически делается внутренний вызов из Start), но в дальнейшем предполагается запрет начала следующего сеанса без сброса предыдущего.

Совместимость: в версиях **NDemia CashMatic** ранее **v.2.7.2** метод Reset вызывается неявно (автоматически) при закрытии окна приложения **NDemia CashMatic KioskBrowser**.

Метод `Start` обозначает начало платёжного сеанса (прикладной скрипт должен вызвать его явно). При этом создаётся начальный набор переменных сеанса:

Имя переменной	Значение	Комментарий
AppSession	номер запуска приложения **NDemia CashMatic KioskBrowser**	счётчик, считается от установки **NDemia CashMatic**, значение увеличивается на единицу при каждом запуске приложения.
PaymentSession	номер текущего платежа	счётчик, считается от установки **NDemia CashMatic**, значение увеличивается на единицу при каждом вызове метода **Start**, однако новое значение запоминается только в случае, если в платёжном сеансе включался приём купюр.
UserSession	номер сеанса приёма платежа	счётчик, считается от установки **NDemia CashMatic**, значение увеличивается на единицу при каждом вызове метода **Start**.

Другими словами: `AppSession` показывает, сколько раз запускался **NDemia CashMatic KioskBrowser**, `UserSession` показывает, сколько раз пользователи начинали платёжные сеансы, `PaymentSession` показывает, сколько раз начинался приём наличных денег.
В процессе эксплуатации терминала нормальным соотношением является:

```
AppSession <= PaymentSession <= UserSession
```

Начальный набор переменных сеанса создаётся до вызова метода `PreStart` интерфейса компонентов расширения `ICashMaticSessionExtender`.

Совместимость: в версиях **NDemia CashMatic** ранее **2.6.0** из перечисленных выше создаётся только переменная `UserSession`, её значение доступно компонентам расширения, начиная с вызова метода `PostStart` интерфейса компонентов расширения `ICashMaticSessionExtender`.

Компоненты расширения могут по событию `PreStart` создать свои дополнительные значения, которые будут впоследствии доступны прикладным скриптам.
Метод `Finish` обозначает конец платёжного сеанса (программист скрипта должен вызвать его явно). В текущей версии это выражается только в соответствующем оповещении компонентов расширения, однако в будущем предполагается запрет каких-либо дальнейших изменений объекта до сброса.
Метод `Finish` обозначает конец платёжного сеанса. В текущей версии это выражается только в соответствующем оповещении компонентов расширения, однако в будущем предполагается запрет каких-либо дальнейших изменений объекта CashMaticSession до сброса (`Reset`).
Предполагается, что хотя бы один из компонентов расширения должен по событиям `PreFinish`/`PostFinish` зарегистрировать платёж (записать в отчётный файл, передать на сервер и т.п.). **NDemia CashMatic KioskBrowser** сам по себе не выполняет зачисление или регистрацию платежа каким-либо образом — это должен сделать либо прикладной скрипт, либо компонент расширения.
Однако механизм платёжных сеансов **NDemia CashMatic** гарантирует следующее: если в текущем платёжном сеансе вызывался метод `Start`, то метод `Finish` будет вызван обязательно. Если прикладной программист не сделает этого вызова явно, будет сделан неявный вызов из `Reset`. Таким образом, `Reset` - это не просто сброс в исходное состояние, отменяющий текущий контекст - для начавшегося сеанса всегда гарантируется завершение.
Рекомендуется вызывать метод `Finish` явно - в текущей версии это не обязательно (автоматически делается внутренний вызов из `Reset`), но в дальнейшем предполагается запрет сброса платёжного сеанса без явного завершения.
Обратите внимание, при определённых обстоятельствах может сложиться целая цепочка неявных вызовов: если прикладной скрипт каждый раз вызывает только `Start`, то `Start` очередного сеанса вызывает неявный `Reset` предыдущего сеанса, а `Reset` неявно вызывает `Finish` предыдущего сеанса.
После `Finish` все данные платежа остаются ещё актуальными, прикладные скрипты и компоненты расширения могут проанализировать, чем закончился платёжный сеанс, были ли обработаны результаты

платежа, переданы ли данные платежа на сервер и т.п.
После `Reset` - никакой информации от предыдущего платежа не остаётся, можно начинать ожидание следующего пользователя-плательщика, который своими действиями откроет новый платёж.
После `Start` - ожидается вызов `Finish`, в обязательном порядке. Если он будет пропущен (не будет сделан прикладным скриптом или компонентом расширения до следующего вызова `Start` или `Reset`), то этот вызов будет выполен автоматически (неявно). В любом случае соответствующим образом уведомляются компоненты расширения (через методы `PreFinish`/`PostFinish` интерфейса `ICashMaticSessionExtender`).

ВАЖНО: программисту следует с большим вниманием отнестись к местам вызова `Reset`, `Start`, `Finish`, поскольку при каких-либо ошибках в этой части есть риск попадания данных предыдущего платежа в следующий сеанс, вплоть до повторного зачисления предыдущего платежа. Кроме того, следует соотносить место сброса сеанса с возможностью возникновения отложенного отказа, т.е. неисправности при незавершённом платёжном сеансе с уже принятыми наличными деньгами (см. Модель обработки отказов).

8.1 Объект CashMaticSession(CashMatic.Session), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
AddProp(PropName, PropValue)	Пустой	*Метод*	Создаёт (или изменяет) переменную платёжного сеанса с именем **PropName** и значением **PropValue**
Event(EventName)	Пустой	*Метод*	Вызывает событие **EventName** в контексте платёжного сеанса. Таким образом скрипт может обратиться к компоненту расширения (хост-приложение вызовет компоненты расширения через **ICashMaticSessionExtender::OnEvent** с указанием **EventName**)
GetProp(PropName)	Строка	*Метод*	Получение значения переменной платёжного сеанса с именем **PropName** (несуществующие переменные возвращаются как пустые строки)
Finish()	Пустой	*Метод*	Конец платёжного сеанса (хост-приложение вызывает компоненты расширения через **ICashMaticSessionExtender::PreFinish**/**ICashMaticSessionExtender::PostFinish**). Конец платёжного сеанса не означает сброса сеанса - данные сеанса остаются валидными, отложенный отказ остаётся отложенным.
Reset()	Пустой	*Метод*	Сброс платёжного сеанса (хост-приложение вызывает компоненты расширения через **ICashMaticSessionExtender::PreReset**/**ICashMaticSessionExtender::PostReset**). Сброс удаляет все данные (переменные) платёжного сеанса.
Start()	Пустой	*Метод*	Начало платёжного сеанса. (хост-приложение вызывает компоненты расширения через **ICashMaticSessionExtender::PreStart**/**ICashMaticSessionExtender::PostStart**).
Text	Строка	*Свойство, только чтение*	Отладочный механизм, возвращает строку, в читаемой форме содержащую все переменные сеанса.

8.2 Компонент расширения для CashMaticSession

Для подключения расширений объекта CashMaticSession компонент расширения должен реализовывать интерфейс ICashMaticSessionExtender или ICashMaticSessionExtender2.

8.2.1 Интерфейс ICashMaticSessionExtender

см. также: Компоненты расширения CashMaticSession ICashMaticSessionExtender2

Для подключения расширений объекта CashMaticSession компонент расширения должен реализовывать интерфейс ICashMaticSessionExtender.
(Дополнительные возможности даёт поддержка интерфейса ICashMaticSessionExtender2).
В синтаксисе C++ интерфейс ICashMaticSessionExtender определяется так:

```
class ICashMaticSessionExtender : public IDispatch
{
        public:
                virtual void CALLBACK
                        PreReset(ICashMaticSession* pSession) = 0;

                virtual void CALLBACK
                        PostReset(ICashMaticSession* pSession) = 0;

                virtual void CALLBACK
                        PreStart(ICashMaticSession* pSession) = 0;

                virtual void CALLBACK
                        PostStart(ICashMaticSession* pSession) = 0;

                virtual void CALLBACK OnEvent(
                                ICashMaticSession* pSession,
                                BSTR EventName
                        ) = 0;

                virtual void CALLBACK
                        PreFinish(ICashMaticSession* pSession) = 0;

                virtual void CALLBACK
                        PostFinish(ICashMaticSession* pSession) = 0;
};

// {02A6987B-E3F2-451c-A81D-35D74CCDE78E}
static const IID IID_ICashMaticSessionExtender = { 0x2a6987b, 0xe3f2,
        0x451c, { 0xa8, 0x1d, 0x35, 0xd7, 0x4c, 0xcd, 0xe7, 0x8e } };
```

Метод ICashMaticSessionExtender::PreReset(ICashMaticSession* pSession) вызывается перед сбросом сеанса. Переменные прошлого сеанса ещё существуют, но после этого вызова больше существовать не будут.

Метод ICashMaticSessionExtender::PostReset(ICashMaticSession* pSession) вызывается после сброса сеанса. Переменных прошлого сеанса уже не существует, началось создание переменных нового сеанса, интерактивного пользователя пока нет.

Метод ICashMaticSessionExtender::PreStart(ICashMaticSession* pSession) вызывается перед началом сеанса (появился интерактивный пользователь). Переменных прошлого сеанса уже не существует, создание переменных нового сеанса ещё не завершено.

Метод ICashMaticSessionExtender::PostStart(ICashMaticSession* pSession) вызывается после начала сеанса. Создание начального набора переменных нового сеанса уже завершено.

Метод ICashMaticSessionExtender::OnEvent(ICashMaticSession* pSession, BSTR EventName) вызывается в результате вызова метода Event объекта CashMaticSession, что доступно скриптам HTML-документа, например:

```
CashMatic.Session.Event("MyEventName");
```
(таким образом можно связать скрипт с компонентом расширения, который существует в контексте хост-приложения, и, соответственно, вне HTML-документа)

Метод `ICashMaticSessionExtender::PreFinish(ICashMaticSession* pSession)` вызывается перед завершением сеанса. Изменение переменных сеанса разрешено, вызов `CashMatic.Session.Event` разрешён.

Метод `ICashMaticSessionExtender::PostFinish(ICashMaticSession* pSession)` вызывается после завершения сеанса. Компоненты расширения и скрипты далее не должны изменять переменные сеанса и вызывать `CashMatic.Session.Event` (текущая версия этому не противодействует, но предполагается, что в дальнейшем это будет запрещено на уровне COM-ошибки)

8.2.2 Интерфейс ICashMaticSessionExtender2

см. также: Компоненты расширения CashMaticSession ICashMaticSessionExtender

Интерфейс `ICashMaticSessionExtender2` является расширенной версией интерфейса `ICashMaticSessionExtender` (добавлен метод `OnAddProp`).
В синтаксисе C++ интерфейс `ICashMaticSessionExtender2` определяется так:

```
class ICashMaticSessionExtender2 : public ICashMaticSessionExtender
{
	public:
		virtual HRESULT STDMETHODCALLTYPE OnAddProp(
			ICashMaticSession* pSession,
			BSTR PropName,
			BSTR PropValue
		) = 0;
};

// {00F6059F-CF7A-4709-870C-DFAD8ECBF2A3}
static const IID IID_ICashMaticSessionExtender2 = { 0xf6059f, 0xcf7a,
	0x4709, { 0x87, 0xc, 0xdf, 0xad, 0x8e, 0xcb, 0xf2, 0xa3 } };
```

Метод `ICashMaticSessionExtender2::OnAddProp(ICashMaticSession* pSession, BSTR PropName, BSTR PropValue)` вызывается при добавлении или изменении переменной платёжного сеанса, т.е. при любом вызове метода `CashMatic.Session.AddProp()`.

Аргументы:

- `pSession` - интерфейс `ICashMaticSession` (см. объект `CashMaticSession`);
- `PropName` - имя переменной платёжного сеанса;
- `PropValue` - значение переменной платёжного сеанса.

Если в **NDemia CashMatic KioskBrowser** установлено несколько (более одного) компонентов расширения, реализующих интерфейс `ICashMaticSessionExtender2`, то метод `OnAddProp` вызывается у каждого из них, по очереди.
Код компонента расширения, реализующий метод `OnAddProp`, должен рассматривать переданное значение `PropValue` как текущее (новое, изменившееся) значение переменной `PropName`, и не должен получать значение переменной `PropName` через `CashMatic.Session.GetProp(PropName)`.

Совместимость: Интерфейс `ICashMaticSessionExtender2` поддерживается в **NDemia CashMatic**, начиная с версии **2.7.0**.

8.3 Интерфейс ICashMaticSession

Компоненты расширения могут обращаться к объекту CashMaticSession через интерфейс ICashMaticSession (компонент расширения при инициализации получает IDispatch* на CashMatic, из него можно получить свойство Session (**dispid 102**) как IDispatch*, а его значение уже преобразовать в ICashMaticSession*)

В синтаксисе C++ интерфейс ICashMaticSession определяется так:

```
class ICashMaticSession : public IDispatch
{
        public:
                virtual void CALLBACK Reset() = 0;

                virtual void CALLBACK Start() = 0;

                virtual void CALLBACK Event(BSTR EventName) = 0;

                virtual void CALLBACK Finish() = 0;

                virtual void CALLBACK AddProp(
                        BSTR PropName,
                        BSTR PropValue
                ) = 0;

                virtual BSTR CALLBACK GetProp(BSTR PropName) = 0;

                virtual HRESULT CALLBACK Text(BSTR* pRetVal) = 0;

                virtual unsigned CALLBACK SetExtender(
                        ICashMaticSessionExtender* pSessionExtender
                ) = 0;

                virtual BOOL CALLBACK RemoveExtender(unsigned) = 0;
};

// {0011C3F9-DF3C-46bb-AFE9-608ECCC6DC5E}
static const IID IID_ICashMaticSession = { 0x11c3f9, 0xdf3c, 0x46bb,
        { 0xaf, 0xe9, 0x60, 0x8e, 0xcc, 0xc6, 0xdc, 0x5e } };
```

Методы Reset, Start, Event, Finish, AddProp, GetProp - см. описание объекта CashMaticSession.

Метод

```
unsigned CALLBACK SetExtender(ICashMaticSessionExtender*);
```

устанавливает компонент расширения для CashMatic.Session (возвращается ненулевой идентификатор компонента, который впоследствии может использоваться для его удаления через метод RemoveExtender). Компонент расширения остаётся установленным до завершения работы программы **NDemia CashMatic KioskBrowser** или до явного вызова RemoveExtender.

> **Совместимость:** в версиях **NDemia CashMatic** ранее **2.6.0** может возвращаться нулевой идентификатор в штатных условиях, т.е. нулевой идентификатор не должен рассматриваться как признак ошибки.

После вызова вызывающий код может освободить ссылку на интерфейс ICashMaticSessionExtender.

> **Совместимость:** в версиях **NDemia CashMatic** ранее **2.7.0** объект CashMaticSession не добавляет собственную ссылку на компонент расширения, поэтому нельзя освобождать ссылку после вызова **SetExtender**.

Метод

```
BOOL CALLBACK RemoveExtender(unsigned);
```

удаляет компонент расширения (в параметре нужно указать идентификатор компонента, возвращённый методом `SetExtender`);

возвращает **`TRUE`** в случае успеха, **`FALSE`** в случае, если компонент расширения с таким идентификатором не установлен.

Совместимость: не рекомендуется использовать в версиях **NDemia CashMatic** ранее **2.7.2** при одновременной работе нескольких компонентов расширения (во многих случаях может приводить к аварийному завершению программы).

9 Объект CashMaticTerminal

Объект `CashMaticTerminal` (`CashMatic.Terminal`) отвечает за контекст работы терминала в целом (вне связи с отдельными сеансами приёма платежа). В текущей реализации `CashMaticTerminal` является основным объектным интерфейсом Модели обработки отказов.

9.1 Объект CashMaticTerminal(CashMatic.Terminal), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
Event(EventName)	Пустой	*Метод*	Вызывает событие **EventName** в контексте терминала. Таким образом скрипт может обратиться к компоненту расширения (хост-приложение вызовет компоненты расширения через **ICashMaticTerminalExtender::OnEvent** с указанием **EventName**)
Failure	Логический	*Свойство, только чтение*	Возвращает истинное значение, если в программе зафиксирован отказ, требующий перехода на страницу неисправности (подробнее см. Модель обработки отказов)
Name	Строка	*Свойство, только чтение*	Возвращает имя терминала по данным настройки **NDemia CashMatic**
ServiceVersionString	Строка	*Свойство, только чтение*	Возвращает версию службы **NDemia CashMatic Kiosk**. Если служба не используется, то возвращается пустая строка. **Совместимость:** не использовать в версиях **NDemia CashMatic** ранее **2.6.0**
SetFailState(FailName, State)	Пустой	*Метод*	Устанавливает отказу **FailName** (строка) состояние **State** (логическое). Крайне не рекомендуется принудительно устанавливать состояния предопредённых отказов (см. Модель обработки отказов). Нормально может использоваться для установки отказов, определённых прикладным программистом.

9.2 Компонент расширения для CashMaticTerminal

Для подключения расширений объекта `CashMaticTerminal` компонент расширения должен поддерживать интерфейс `ICashMaticTerminalExtender` или `ICashMaticTerminalExtender2`.

9.2.1 Интерфейс ICashMaticTerminalExtender

Для подключения расширений объекта `CashMaticTerminal` компонент расширения должен поддерживать интерфейс `ICashMaticTerminalExtender`.
(Дополнительные возможности даёт поддержка интерфейса `ICashMaticTerminalExtender2`.)
В синтаксисе C++ интерфейс `ICashMaticTerminalExtender` определяется так:

```
class ICashMaticTerminalExtender : public IDispatch
{
        public:
                virtual void CALLBACK
                        OnGuiStartup(ICashMaticTerminal* pTerminal) = 0;

                virtual void CALLBACK
                        OnGuiExit(ICashMaticTerminal* pTerminal) = 0;

                virtual void CALLBACK OnEvent(
                        ICashMaticTerminal* pTerminal,
                        BSTR EventName
                ) = 0;

                virtual void CALLBACK OnFailure(
                        ICashMaticTerminal* pTerminal,
                        BSTR FailName,
                        BOOL FailState
                ) = 0;

                virtual void CALLBACK
                        PreCashCollect(ICashMaticTerminal* pTerminal) = 0;

                virtual void CALLBACK
                        PostCashCollect(ICashMaticTerminal* pTerminal) = 0;

                virtual void CALLBACK
                        PreZReport(ICashMaticTerminal* pTerminal) = 0;

                virtual void CALLBACK
                        PostZReport(ICashMaticTerminal* pTerminal) = 0;
};

// {00735A83-6101-49ae-B2E5-77BA59B3D6A9}
static const IID IID_ICashMaticTerminalExtender = { 0x735a83, 0x6101,
        0x49ae, { 0xb2, 0xe5, 0x77, 0xba, 0x59, 0xb3, 0xd6, 0xa9 } };
```

Метод `ICashMaticTerminalExtender::OnGuiStartup(ICashMaticTerminal* pTerminal)` вызывается при запуске **NDemia CashMatic KioskBrowser**. Вызов этого метода свидетельствует о нормальной установке и инициализации компонента расширения.

Метод `ICashMaticTerminalExtender::OnGuiExit(ICashMaticTerminal* pTerminal)` вызывается при завершении работы **NDemia CashMatic KioskBrowser**. Гарантируется, что это последний вызов компонента расширения перед завершением работы (точнее: гарантируется, что после завершения `OnGuiExit` всех установленных компонентов расширения работа приложения будет завершена без каких-либо дополнительных вызовов прикладного кода).

Метод `ICashMaticTerminalExtender::OnEvent(ICashMaticTerminal* pTerminal, BSTR EventName)` вызывается в результате вызова метода `Event` объекта `CashMaticTerminal`, что доступно скриптам HTML-документа, например:

```
CashMatic.Terminal.Event("MyEventName");
```

(таким образом можно связать скрипт с компонентом расширения, который существует в контексте хост-приложения, и, соответственно, вне HTML-документа)

Метод `ICashMaticTerminalExtender::OnFailure(ICashMaticTerminal* pTerminal, BSTR FailName, BOOL FailState)` вызывается в результате возникновения (`FailState` - истина) или устранения (`FailState` - ложь) именованного отказа `FailName` (см. Модель обработки отказов). Обработка отказа или восстановления в данной точке не должна зависеть от режима серьёзности отказа и от режима отложенного отказа.

Метод `ICashMaticTerminalExtender::PreCashCollect(ICashMaticTerminal* pTerminal)` вызывается при снятии (удалении) стекера (кассы) купюроприёмника. Здесь компонент расширения может сделать переход на страницу инкассации (через `CashMatic.Terminal.Navigate`), которая должна быть зарегистрирована как специальный URL с именем `collect` (см. Специальные URL). Любые другие адреса переходов навигации блокируются до восстановления купюроприёмника.

Метод `ICashMaticTerminalExtender::PostCashCollect(ICashMaticTerminal* pTerminal)` вызывается при восстановлении стекера (кассы) купюроприёмника после снятия. Здесь компонент расширения может сделать переход на любой URL-адрес (через `CashMatic.Terminal.Navigate`)

Метод `ICashMaticTerminalExtender::PreZReport(ICashMaticTerminal* pTerminal)` вызывается перед печатью отчёта с гашением на фискальном регистраторе. Кассовая смена ещё не закрыта.

Метод `ICashMaticTerminalExtender::PostZReport(ICashMaticTerminal* pTerminal)` вызывается после печати отчёта с гашением на фискальном регистраторе. Кассовая смена уже закрыта.

9.2.2 Интерфейс ICashMaticTerminalExtender2

Интерфейс `ICashMaticTerminalExtender2` является расширенной версией интерфейса `ICashMaticTerminalExtender` (добавлен метод `OnScriptError`).
В синтаксисе C++ интерфейс `ICashMaticTerminalExtender2` определяется так:

```
class ICashMaticTerminalExtender2 : public ICashMaticTerminalExtender
{
        public:
                virtual HRESULT STDMETHODCALLTYPE OnScriptError(
                        ICashMaticTerminal* pTerminal,
                        IDispatch* pWindow,
                        BSTR ErrorMessage,
                        BSTR URL,
                        UINT Line,
                        VARIANTARG *pRetVal
                ) = 0;
};

// {0ACA9168-ED81-41bc-AA16-F575D35EF8FE}
static const IID IID_ICashMaticTerminalExtender2 = { 0xaca9168, 0xed81,
        0x41bc, { 0xaa, 0x16, 0xf5, 0x75, 0xd3, 0x5e, 0xf8, 0xfe } };
```

Метод `ICashMaticTerminalExtender2::OnScriptError(ICashMaticTerminal* pTerminal, IDispatch* pWindow, BSTR ErrorMessage, BSTR URL, UINT Line, VARIANTARG *pRetVal)` вызывается при ошибке прикладного скрипта.

Аргументы:

- `pTerminal` - интерфейс `ICashMaticTerminal` (см. объект `CashMaticTerminal`);
- `pWindow` - объект `window` - окно, в котором произошла ошибка (см. документацию по объектам **Internet Explorer**), компонент расширения не должен освобождать ссылку на этот объект;
- `ErrorMessage` - текст соообщения об ошибке;
- `URL` - адрес HTML-страницы, в которой произошла ошибка;
- `Line` - номер строки исходного HTML-кода, на которой произошла ошибка;
- `*pRetVal` - возвращаемое логическое значение. Если метод возвращает истинное значение, то диалог ошибки браузера не показывается (не выполняется обычная обработка ошибки **Internet Explorer**). Если метод возвращает ложное значение, то выполняется обычная обработка ошибки **Internet Explorer**.

Если в **NDemia CashMatic KioskBrowser** установлено несколько (более одного) компонентов расширения, реализующих интерфейс `ICashMaticTerminalExtender2`, то метод `OnScriptError` вызывается у каждого из них, по очереди. Если хотя бы один из них вернёт истинное значение, то обычная обработка ошибки не выполняется.

Прикладной скрипт, устанавливающий собственную обработку **`window.onerror`**, перехватывает обработку ошибок **NDemia CashMatic KioskBrowser**, необходимую для срабатывания метода `OnScriptError`.
В этом случае прикладной скрипт должен сохранить значение свойства **`window.onerror`** на момент запуска скрипта, и в случае вызова события **`window.onerror`** использовать сохранённое значение для вызова обработчика **NDemia CashMatic KioskBrowser** - см. Обработка событий window.onerror.

Совместимость: Интерфейс `ICashMaticTerminalExtender2` поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**.

9.3 Интерфейс ICashMaticTerminal

Компоненты расширения могут обращаться к объекту `CashMaticTerminal` через интерфейс `ICashMaticTerminal` (компонент расширения при инициализации получает `IDispatch*` на `CashMatic`, из него можно получить свойство `Terminal` (**`dispid 104`**) как `IDispatch*`, а его значение уже преобразовать в `ICashMaticTerminal*`).

В синтаксисе C++ интерфейс `ICashMaticTerminal` определяется так:

```
class ICashMaticTerminal : public IDispatch
{
        public:
                virtual HRESULT CALLBACK Name(BSTR* pRetVal) = 0;

                virtual HRESULT CALLBACK
                        ServiceVersionString(BSTR* pRetVal) = 0;

                virtual void CALLBACK Event(BSTR EventName) = 0;

                virtual HRESULT CALLBACK Failure(BOOL* pRetVal) = 0;

                virtual void CALLBACK SetFailState(
                        BSTR FailName,
                        BOOL State
                ) = 0;

                virtual unsigned CALLBACK SetExtender(
                        ICashMaticTerminalExtender* pTerminalExtender
                ) = 0;

                virtual BOOL CALLBACK RemoveExtender(unsigned) = 0;
};

// {01F1C52B-B434-4c9c-AFC9-3474EFF712E7}
static const IID IID_ICashMaticTerminal = { 0x1f1c52b, 0xb434, 0x4c9c,
        { 0xaf, 0xc9, 0x34, 0x74, 0xef, 0xf7, 0x12, 0xe7 } };
```

Методы `Name`, `ServiceVersionString`, `Event`, `Failure`, `SetFailState` - см. описание объекта `CashMaticTerminal`.

Метод

```
unsigned CALLBACK SetExtender(ICashMaticTerminalExtender*);
```

устанавливает компонент расширения для `CashMatic.Terminal` (возвращается ненулевой идентификатор компонента, который впоследствии может использоваться для его удаления через метод `RemoveExtender`). Компонент расширения остаётся установленным до завершения работы программы **NDemia CashMatic KioskBrowser** или до явного вызова `RemoveExtender`.

> **Совместимость:** в версиях **NDemia CashMatic** ранее **2.6.0** может возвращаться нулевой идентификатор в штатных условиях, т.е. нулевой идентификатор не должен рассматриваться как признак ошибки.

После вызова вызывающий код может освободить ссылку на интерфейс `ICashMaticTerminalExtender`.

> **Совместимость:** в версиях **NDemia CashMatic** ранее **2.7.0** объект `CashMaticTerminal` не добавляет собственную ссылку на компонент расширения, поэтому нельзя освобождать ссылку после вызова **`SetExtender`**.

Метод

```
BOOL CALLBACK RemoveExtender(unsigned);
```

удаляет компонент расширения (в параметре нужно указать идентификатор компонента, возвращённый методом `SetExtender`)

Возвращает **`TRUE`** в случае успеха, **`FALSE`** в случае, если компонент расширения с таким идентификатором не установлен.

Совместимость: не рекомендуется использовать в версиях **NDemia CashMatic** ранее **2.7.2** при одновременной работе нескольких компонентов расширения (во многих случаях может приводить к аварийному завершению программы).

10 Печать

Одной из главных функций программного продукта **NDemia CashMatic** является выпуск печатных документов, подтверждающих приём платежа от пользователя-плательщика - чеков, квитанций, и т.п. (обобщённо выходные печатные документы называются чеками) При этом предоставляемые программным продуктом средства позволяют разработчику платёжной системы абстрагироваться от конкретики применяемого на терминале печатающего устройства - это может быть чековый принтер, фискальный регистратор и даже просто текстовый файл.
Для веб-программиста печать чеков выполняется объектом `CashMaticPrint` (`CashMatic.Print`). Более подробно см. Объекты печати.
`CashMaticPrint` и другие объекты печати предоставляют набор средств, пользуясь которыми прикладная программа должна сформировать собственно выводимый на печать текст. Более подробно см. Формирование текста.
Важным механизмом организации выходных документов являются шаблоны печати. Общая идея состоит в том, что прикладная программа (java-скрипт на HTML-странице) поставляет через объекты печати хост-приложению (**NDemia CashMatic KioskBrowser**) максимум информации о содержании печатного документа (чека) и минимум сведений о форматировании и параметрах печати. Параметры печати настраиваются по месту, на конкретном терминале, через Панель управления **NDemia CashMatic**. Форматирование документа (т.е. что, где и как должно быть конкретно напечатано на чеке) задаётся шаблоном печати. Шаблоны печати могут легко редактироваться в текстовом формате, без изменения прикладной программы. Шаблоны печати могут различаться на разных терминалах, работающих в одной платёжной системе, и эти различия будут почти полностью скрыты от java-скрипта. Это повышает универсальность прикладных скриптов, одновременно предоставляя довольно большую гибкость в настройке чека на конкретном оборудовании. Более подробно см. Шаблоны печати.

10.1 Объекты печати

За всё, связанное с печатью, принтером, фискальным регистратором отвечает объект `CashMaticPrint`. Основные функции объекта `CashMaticPrint`:

- проверка наличия бумаги в принтере (метод `CheckPaper()`);
- формирование текста для печати (свойства `Receipt`, `Text`);
- выдача текста на печать (метод `PrintOut()`);
- предоставление сведений о возможностях и настройках оборудования и программного обеспечения подсистемы печати (свойства `Caps`, `Options`);
- идентификация оборудования (свойство `SerialNumber`);
- статус фискализации (свойства `LastFiscalResult`, `FiscalSaleTotal`, метод `CreateFiscalSessionSummary()`);
- печать отчётов фискального регистратора (методы `XReport()`, `ZReport()`);
- дополнительное управление отрезкой и перемещением напечатанного чека (методы `Cut()`, `Eject()`, `Retract()`).

Формирование текста для печати возможно в следующих режимах:

- текст без шаблона;
- фискальный чек без шаблона;
- чек по шаблону.

Объект `CashMaticPrint` существует в контексте работы хост-приложения, а не HTML-документа, скрипт может лишь получить ссылку на этот объект, но не может его создать или уничтожить. Перед каждой печатью требуется явный сброс объекта методом `Reset()`. С платёжным сеансом объект `CashMaticPrint` в общем не связан, допускается многократная печать в пределах одного сеанса. При отсутствии явного сброса допускается переход чека в следующий платёжный сеанс, хотя это само по себе

неправильно и в будущих версиях вероятнее всего будет запрещено.

Опросить возможности подсистемы печати, определяемые установленным оборудованием и настройками печати **NDemia CashMatic**, можно через свойства объекта `CashMaticPrint`:

- `Caps` - возвращает объект `CashMaticPrintCaps`, через который можно получить возможности оборудования.
- `Options` - возвращает объект `CashMaticPrintOptions`, через который можно получить основные настройки печати, заданные в системном реестре через Панель управления **NDemia CashMatic** и/или команду `CONFIG`.

Свойства объекта `CashMaticPrintCaps` (доступен как свойство `Caps` объекта `CashMaticPrint`)

- `Cutter` - логическое значение, показывающее наличие отрезчика чеков;
- `Fiscal` - логическое значение, показывающее, установлен ли драйвер фискального регистратора.

Свойства объекта `CashMaticPrintOptions` (доступен как свойство `Options` объекта `CashMaticPrint`):

- `CharFormatting` - логическое значение, показывающее, включено ли использование кодов форматирования (см. Форматирование шрифта);
- `FileMode` - логическое значение, истина, если вывод печати перенаправлен в текстовый файл;
- `Fiscal` - логическое значение, истина, если используется фискальный регистратор;
- `PrinterMode` - логическое значение, истина, если текущим устройством печати является принтер.

За формирование чека продажи отвечает объект `CashMaticReceipt`, доступен как свойство `Receipt` объекта `CashMaticPrint`.
Наиболее важными свойствами объекта `CashMaticReceipt` являются:

- `SaleItems` - список оплачиваемых товаров и услуг, реализуется объектом `CashMaticSaleItems`. Требуется заполнение `SaleItems` в каждом чеке (даже если принимается только один платёж по одной единой для всех пользователей услуге);
- `Cash` - сумма внесённых наличных (указывать нужно явно, **не** вычисляется по суммарной стоимости `SaleItems`);
- `Template` - имя шаблона чека (указывать нужно явно, в каждом чеке) - см. Шаблоны печати).

Необязательные для заполнения свойства объекта `CashMaticReceipt`:

- `Id` - идентификатор чека (используется, только если в шаблоне чека есть соответствующая ссылка);
- `Code` - код чека, код документа (используется, только если в шаблоне чека есть соответствующая ссылка) - рекомендуемое (но не обязательное) свойство для значения штрихкода чека;
- `Title` - заголовок чека (используется, только если в шаблоне чека есть соответствующая ссылка);
- `Fiscal` - выполнять ли фискализацию продажи (по умолчанию используется значение из настроек программы).

Объект `CashMaticReceipt` предоставляет ряд информационных свойств, которые могут использоваться скриптом для получения дополнительных сведений по ходу оформления продажи:

- `Date` - дата печати чека;
- `Time` - время печати чека;
- `Terminal` - имя терминала, использованное при печати чека;
- `Total` - итоговая стоимость чека по списку `SaleItems`;
- `Change` - сумма сдачи как разность `Cash - Total` (только если `Cash > Total`);
- `Debt` - сумма долга как разность `Total - Cash` (только если `Cash < Total`).

Список продажи содержит сведения о номенклатуре, ценах и количествах товаров/услуг, оплаченных пользователем. Реализуется объектом `CashMaticSaleItems`, который доступен как свойство `SaleItems` объекта `CashMaticReceipt`.
Объект `CashMaticSaleItems` имеет методы, типичные как для COM-коллекции: `Add()`, `Item()`, `Remove()`, `Count()`, так и для javascript-коллекции: `length`. Кроме того, в режиме "фискальный чек без шаблона" есть возможность вставить текст для печати над и под всем списком товаров/услуг (свойства `TextAbove`/`TextBelow`). В остальных режимах запись этих свойств игнорируется, а при чтении они содержат текст до (`TextAbove`) и после (`TextBelow`) списка продажи (`SaleItems`), в том виде, как он сформировался по шаблону.

Список `CashMaticSaleItems` содержит элементы типа `CashMaticSaleItem`.
Важными свойствами, требующими обязательного заполнения, являются:

- `Name` - наименование товара/услуги;
- `Price` - цена единицы товара/услуги;
- `Quantity` - количество оплачиваемых единиц товара/услуги.

Кроме того, есть возможность вставить текст для печати над и под строкой товара/услуги (свойства `TextAbove`/`TextBelow`)

Объект `CashMaticFiscalSessionSummary` (доступен как результат вызова метода `CreateFiscalSessionSummary()` объекта `CashMaticPrint`) представляет собой сводку текущих итогов продаж по кассовой смене (используется только на фискальном регистраторе).
Объект предоставляет прикладному программисту следующие сведения (свойства, доступные для чтения):

- `TimeStamp` - дата и время получения сводки по часам фискального регистратора;
- `SessionNumber` - номер кассовой смены, по которой получена сводка;
- `Closed` - признак закрытой кассовой смены (по которой уже снят отчёт с гашением);
- `SaleReceiptCount` - количество напечатанных чеков продажи по кассовой смене;
- `SaleTotal` - сумма продаж по кассовой смене, в рублях.

Свойства объекта `CashMaticFiscalSessionSummary` доступны только для чтения, поскольку он выражает некоторое зафиксированное состояние кассовой смены, которое не подлежит программным изменениям.

10.1.1 Объект CashMaticPrint(CashMatic.Print), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
Caps	Объект **CashMaticPrintCaps**	*Свойство, только чтение*	Объект, показывающий возможности установленного оборудования печати.
CheckPaper()	Логический	*Метод*	Проверка наличия бумаги в принтере. Внутренне вызывает соответствующее изменение состояние отказов **Printer**, **PaperEnd**, **NearPaperEnd** (см. Модель обработки отказов).
CreateFiscalSessionSummary()	Объект **CashMaticFiscalSessionSummary**	*Метод*	Получение сводки по кассовой смене (используется только на фискальном регистраторе). **Совместимость:** поддерживается, начиная с версии **2.6.0**
Cut()	Логический	*Метод*	Отрезать чек. Возвращается истинный результат при успешном выполнении. Поддерживается только при наличии и правильной настройке соответствующего оборудования. **Совместимость:** поддерживается, начиная с версии **2.6.0**
Eject()	Логический	*Метод*	Вытолкнуть чек из механизма удержания чека. Возвращается истинный результат при успешном выполнении. Поддерживается только при наличии и правильной настройке соответствующего оборудования. **Совместимость:** поддерживается, начиная с версии **2.6.0**
FiscalSaleTotal	CY	*Свойство, только чтение*	Сумма продаж по фискальной памяти (используется только на фискальном регистраторе). **Совместимость:** поддерживается, начиная с версии **2.6.0**

LastFiscalResult	Логический	*Свойство, только чтение*	Признак фискализации последней печати (используется только на фискальном регистраторе). Истинное значение указывает, что в результате предшествующего вызова метода **PrintOut()** в данных фискального регистратора повысилась сумма продаж по кассовой смене. Метод **Reset()** сбрасывает значение. **Совместимость:** поддерживается, начиная с версии **2.6.0**
Options	Объект **CashMaticPrintOptions**	*Свойство, только чтение*	Объект, показывающий настройки подсистемы печати, заданные в Панели управления **NDemia CashMatic**.
PrintOut()	Логический	*Метод, не вызывать после печати* (•)	Печать текущего текста или чека на принтер или фискальный регистратор.
Receipt	Объект **CashMaticReceipt**	*Свойство, только чтение*	Объект текущего чека.
Reset()	Пустой	*Метод*	Общий сброс текущего контекста печати. Все связанные объекты (**Receipt**, **Text**) также сбрасываются. **Совместимость:** в версиях ранее **2.7.1** не сбрасывается свойство **Code** объекта **Receipt**.
Retract()	Логический	*Метод*	Втянуть чек из механизма удержания чека. Возвращается истинный результат при успешном выполнении. Поддерживается только при наличии и правильной настройке соответствующего оборудования. **Совместимость:** поддерживается, начиная с версии **2.6.0**
SerialNumber	Строка	*Свойство, только чтение*	Серийный номер принтера. *В текущей версии поддерживается только для фискального регистратора.* **Совместимость:** поддерживается, начиная с версии **2.6.0**
Text	Строка	*Свойство, чтение/запись, после печати только чтение* (••)	Текст для печати. При записи свойства **Text** объект **Receipt** сбрасывается. Чтение возвращает выводимый текст и в режиме текста, и в режиме чека.
XReport()	Пустой	*Метод*	Отчёт без гашения (используется только на фискальном регистраторе).

ZReport()	Пустой	*Метод*	Отчёт с гашением (используется только на фискальном регистраторе). Компоненты расширения объекта **CashMaticTerminal** отдельно оповещаются об этом событии (см. методы **ICashMaticTerminalExtender::PreZReport** / **ICashMaticTerminalExtender::PostZReport**).

(•) запрещено вызывать метод после выполнения фактической печати чека, иначе произойдёт программная ошибка с сообщением "Уже напечатано".
(••) запрещено изменять значение свойства после выполнения фактической печати чека, иначе произойдёт программная ошибка с сообщением "Уже напечатано".

10.1.2 Объект CashMaticPrintCaps (CashMatic.Print.Caps), свойства

Имя	Тип результата	Тип обращения	Назначение
Cutter	Логический	*Свойство, только чтение*	Истина, если печатное оборудование оснащено отрезчиком чековой ленты. *Текущая реализация: возвращает истину, если в качестве текущего принтера выбран фискальный регистратор, или если в системе команд текущего принтера настроена команда **CUT**.* **Совместимость:** поддерживается, начиная с версии **2.6.2**
Fiscal	Логический	*Свойство, только чтение*	Истина, если на компьютере/терминале установлен драйвер фискального регистратора. (•)

(•) Истинное значение свойства `Fiscal` объекта `CashMatic.Print.Caps` показывает принципиальную возможность выбора фискального регистратора в качестве текущего принтера. Узнать, выбран ли действительно фискальный регистратор как текущий принтер, можно через свойство `Fiscal` объекта `CashMatic.Print.Options`.

10.1.3 Объект CashMaticPrintOptions (CashMatic.Print.Options), свойства

Имя	Тип результата	Тип обращения	Назначение
CharFormatting	Логический	*Свойство, только чтение*	Истина, если включено использование кодов форматирования (см. Форматирование шрифта).
FileMode	Логический	*Свойство, только чтение*	Истина, если вывод печати перенаправлен в текстовый файл.
Fiscal	Логический	*Свойство, только чтение*	Истина, если используется фискальный регистратор. (•)
PrinterMode	Логический	*Свойство, только чтение*	Истина, если текущим устройством печати является принтер.

(•) Истинное значение свойства `Fiscal` объекта `CashMatic.Print.Options` показывает, выбран ли фискальный регистратор в качестве текущего принтера. Принципиальную возможность выбора фискального регистратора можно узнать через свойство `Fiscal` объекта `CashMatic.Print.Caps`. Фискальный режим конкретного чека задаётся свойством `Fiscal` объекта `CashMatic.Print.Receipt`.

10.1.4 Объект CashMaticReceipt(CashMatic.Print.Receipt), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
Cash	CY	*Свойство, чтение/запись, после печати только чтение* (•)	Количество денег, внесённых по чеку (в денежных единицах)
Change	CY	*Свойство, только чтение*	Сумма сдачи как разность `Cash - Total` (только если `Cash > Total`) (в денежных единицах)
Code	Строка	*Свойство, чтение/запись, после печати только чтение* (•)	Код документа (чека). Используется, только если в шаблоне чека есть соответствующая ссылка. *Рекомендуемое (но не обязательное) свойство для штрих-кода чека.* **Совместимость:** поддерживается, начиная с версии **2.6.0** **Совместимость:** сброс методом **Reset()** поддерживается, начиная с версии **2.7.1**
Date	DATE	*Свойство, только чтение после печати* (••)	Дата чека, как она зафиксирована на чеке
Debt	CY	*Свойство, только чтение*	Сумма долга как разность `Total - Cash` (только если `Cash < Total`) (в денежных единицах)
Fiscal	Логический	*Свойство, чтение/запись, после печати только чтение* (•)	Фискальная продажа (используется только на фискальном регистраторе)
Id	Строка	*Свойство, чтение/запись, после печати только чтение* (•)	Идентификатор чека (используется, только если в шаблоне чека есть соответствующая ссылка)
Reset()	Пустой	*Метод*	Сброс контекста чека и всех связанных с ним данных **Совместимость:** в версиях ранее **2.7.1** не сбрасывается свойство **Code**.
SaleItems	Коллекция **CashMatic SaleItems**	*Свойство, только чтение*	Список элементов продажи (товаров и услуг)
Template	Строка	*Свойство, чтение/запись, после печати только чтение* (•)	Имя шаблона чека (указывать нужно явно, в каждом чеке. Формат, синтаксис и размещение шаблонов - см. Шаблоны печати)
Terminal	Строка	*Свойство, только чтение после печати* (••)	Имя терминала, как оно зафиксировано на чеке (по данным настройки **NDemia CashMatic**)
Time	DATE	*Свойство, только чтение после печати* (••)	Время чека, как оно зафиксировано на чеке
Title	Строка	*Свойство, чтение/запись, после печати только чтение* (•)	Заголовок чека (используется, только если в шаблоне чека есть соответствующая ссылка)
Total	CY	*Свойство, только чтение*	Итоговая стоимость чека по данным **SaleItems** (в денежных единицах) (•••)

(•) запрещено изменять значение свойства после выполнения фактической печати чека, иначе произойдёт программная ошибка с сообщением "Уже напечатано".
(••) свойство фиксирует значение именно на момент печати чека, поэтому оно недоступно для изменения, а его чтение до выполнения фактической печати чека вызывает ошибку с сообщением "Ещё не напечатано".
(•••) следует обращать внимание на округление исходных данных, на основании которых вычисляется стоимость чека (цен и количества единиц позиций списка продажи - см. объекты `CashMaticSaleItems` и `CashMaticSaleItem`). При вычислении стоимости чека стоимость каждой строки округляется по формату денег, заданному в настройке региональных параметров Windows (но не более 4х дробных знаков).

10.1.5 Объект CashMaticSaleItems(CashMatic.Print.Receipt.SaleItems), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
Add()	Объект **CashMaticSaleItem**	*Метод*	Добавление нового элемента в список оплачиваемых товаров и услуг
Count()	Число	*Метод*	Число элементов в списке оплачиваемых товаров и услуг (значение аналогично свойству **length)**
Item(index)	Объект **CashMaticSaleItem**	*Метод*	Возвращает элемент списка оплачиваемых товаров и услуг под номером **index** (начиная с нулевого)
length	Число	*Свойство, только чтение*	Число элементов в списке оплачиваемых товаров и услуг (значение аналогично методу **Count**)
Remove(index)	Пустой	*Метод*	Удаляет элемент списка оплачиваемых товаров и услуг под номером **index** (начиная с нулевого)
Reset()	Пустой	*Метод*	Сброс списка и связанных данных (**TextAbove** / **TextBelow**)
TextAbove	Строка	*Свойство, чтение/запись*	Текст для печати перед списком оплачиваемых товаров и услуг
TextBelow	Строка	*Свойство, чтение/запись*	Текст для печати под списком оплачиваемых товаров и услуг

10.1.6 Объект CashMaticSaleItem(элемент коллекции CashMatic.Print.Receipt.SaleItems), свойства и методы

Имя	Тип результата	Тип обращения	Назначение
Name	Строка	*Свойство, чтение/запись*	Наименование товара или услуги
Price	CY	*Свойство, чтение/запись*	Цена единицы товара или услуги(•)
Quantity	Число	*Свойство, чтение/запись*	Количество единиц товара или услуги(•)
Reset()	Пустой	*Метод*	Сброс всех значений
TextAbove	Строка	*Свойство, чтение/запись*	Текст для печати над элементом списка оплачиваемых товаров и услуг
TextBelow	Строка	*Свойство, чтение/запись*	Текст для печати под элементом списка оплачиваемых товаров и услуг

(•) при задании значений `Price` и `Quantity` следует обращать внимание на округление. Эти числа участвуют в расчёте стоимости чека, поэтому использование неокруглённых дробных чисел может приводить к нарастающей погрешности результата. Кроме того, при фискальной печати эти значения передаются фискальному регистратору, при этом претерпевая несколько трансформаций (в ходе которых может произойти изменение разрядности числа с плавающей запятой), расчёт стоимости чека делается самим фискальным регистратором - в целом всё это может привести к расхождению денежных сумм в разных узлах платёжной системы. **NDemia CashMatic** принудительно округляет значения до количества дробных знаков, заданного в настройке региональных параметров Windows (`Price` по формату денег (но не более 4 дробных знаков), `Quantity` по формату числа).

10.1.7 Объект CashMaticFiscalSessionSummary, свойства

Имя	Тип результата	Тип обращения	Назначение
Closed	Логический	*Свойство, только чтение*	Признак закрытой кассовой смены. Истинное значение показывает, что кассовая смена на фискальном регистраторе закрыта (не открыта).
SaleReceiptCount	Число	*Свойство, только чтение*	Количество чеков продажи по кассовой смене
SaleTotal	CY	*Свойство, только чтение*	Сумма продаж по кассовой смене
SessionNumber	Число	*Свойство, только чтение*	Номер кассовой смены
TimeStamp	Дата/время	*Свойство, только чтение*	Дата и время получения сводки, т.е. создания объекта **CashMaticFiscalSessionSummary**, по часам фискального регистратора

Совместимость: объект `CashMaticFiscalSessionSummary` поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

10.2 Формирование текста

При помощи объектов печати прикладной скрипт предоставляет программе **NDemia CashMatic KioskBrowser** информацию о содержании печатного документа (чека).
Формирование текста для печати возможно в следующих режимах:

- текст без шаблона (текст полностью формируется прикладным скриптом)
- фискальный чек без шаблона (текст полностью формируется фискальным регистратором)
- чек по шаблону (текст формируется программой **NDemia CashMatic** заполнением заранее подготовленного шаблона, при печати на фискальном регистраторе часть текста может быть изменена)

Кроме этого, во всех режимах текст может содержать дополнительное форматирование шрифтов - наклон, подчёркивание и т.п. Для фискального регистратора поддерживается также выбор шрифта строки из таблицы шрифтов. Более подробно см. Форматирование шрифта.

Заметим, что наиболее универсальным (и предпочтительным для прикладного скрипта) является режим "чек по шаблону".

10.2.1 Текст без шаблона

Текст без шаблона является самым простым с точки зрения программной логики способом печати (в то же время наименее универсальным и наиболее нагружающим прикладной скрипт особенностями печати на конкретном устройстве).
Нужно просто задать значение свойству `CashMatic.Print.Text` и вызвать метод `CashMatic.Print.PrintOut()`. При этом в строке `Text` могут содержаться и управляющие данные - переносы строк, ESC-команды принтера и т.п.

Пример:
```
CashMatic.Print.Reset();       //начальная установка объекта печати
CashMatic.Print.Text = "Пример прямой печати\r\nВторая строка";
CashMatic.Print.PrintOut();    //собственно печать
```
Следует обратить внимание, что на выводимый текст действуют параметры печати, заданные в Панели управления **NDemia CashMatic** - преобразование кодировки и обработка %-кодов форматирования. Если их отключить, то текст будет выдаваться на печать "один-к-одному", без преобразований, включая двоичные данные и управляющие последовательности.
Если Вы используете печать через `CashMatic.Print.Text`, то Вы не должны выполнять никаких действий с объектом `CashMatic.Print.Receipt` (эти действия либо будут проигнорированы, либо изменят значение `CashMatic.Print.Text`).

10.2.2 Фискальный чек без шаблона

Фискальный регистратор может самостоятельно отформатировать чек, без каких-либо дополнительных указаний со стороны прикладной программы, поэтому использование шаблона чека в общем случае не является обязательным.
Прикладному скрипту при наличии фискального регистратора достаточно заполнить `CashMatic.Print.Receipt.SaleItems`, `CashMatic.Print.Receipt.Cash`, и затем вызвать `CashMatic.Print.PrintOut()`.

Пример:
```
/*
        Печать на фискальном регистраторе чека продажи на 2 позиции:
                - платёж за услугу на 100 руб.
                - 5 конфет по 5 рублей.
        при условии внесения наличных 150 руб.
*/
CashMatic.Print.Reset();       //начальная установка объекта печати

//Добавляем позицию в список продажи:
```

```
Item = CashMatic.Print.Receipt.SaleItems.Add();
Item.Name = "Платёж за услугу";       //название позиции в списке продажи
Item.Price = 100;      //цена товарной единицы
Item.Quantity = 1;     //количество оплаченного товара

//Добавляем позицию в список продажи:
Item = CashMatic.Print.Receipt.SaleItems.Add();
Item.Name = "Конфета"; //название позиции в списке продажи
Item.Price = 5;        //цена товарной единицы
Item.Quantity = 5;     //количество оплаченного товара

//количество внесённых наличных:
CashMatic.Print.Receipt.Cash = 150;

//собственно печать чека:
if(CashMatic.Print.PrintOut())
{
        //при успешной печати:
        SendInfoToServer(
                CashMatic.Print.Receipt.Terminal,
                CashMatic.Print.Receipt.Date,
                CashMatic.Print.Receipt.Time,
                CashMatic.Print.Receipt.Total,
                CashMatic.Print.Receipt.Cash
        );//предполагаемая функция прикладной программы,
        //      отсылающая на сервер данные чека: имя терминала,
        //      дата, время, итоговая сумма, внесённые наличные
        //      - всё в том виде, как было напечатано на чеке.
        //(функция SendInfoToServer не написана, указана
        //      только для иллюстрации)
}
```

Обратите внимание, итоговая стоимость чека составляет 125 руб., внесено 150 руб., фискальный регистратор должен на чеке показать сдачу 25 руб. и сделать соответствующую запись в фискальной памяти.

Дополнительно можно задать значения свойствам `TextAbove` и/или `TextBelow` (текст над/текст под) для каждого элемента (позиции) в списке продажи.
Также можно задать значения свойствам `TextAbove` и/или `TextBelow` (текст над/текст под) всего списка продажи (`CashMatic.Print.Receipt.SaleItems.TextAbove` / `CashMatic.Print.Receipt.SaleItems.TextBelow`) - поддерживается только в режиме фискального чека без шаблона.

Следует обратить внимание, что на выводимый текст действует параметр печати, задаваемый в Панели управления **NDemia CashMatic** - обработка %-кодов форматирования. На самом деле форматирование %-кодами на фискальном регистраторе не поддерживается, обработка состоит в их извлечении из текста (это можно отключить).
Если Вы организуете печать через `CashMatic.Print.Receipt`, то Вы не должны изменять значение `CashMatic.Print.Text`, иначе печать автоматически переключится в режим "текст без шаблона", без фискализации.

Если указанный выше пример будет выполнен не на фискальном регистраторе, а на обычном принтере, то не напечатается совсем ничего - текст должен быть сформирован либо целиком программно (см. текст без шаблона), либо заполнением заранее подготовленного шаблона (см. чек по шаблону)
Аналогично ничего не напечатается даже на фискальном регистраторе, если в исходном тексте примера выше `CashMatic.Print.PrintOut()` вставить строку

```
CashMatic.Print.Receipt.Fiscal = false;
```

поскольку это будет означать нефискальный режим режим печати, т.е. текст чека формируется программно и отдаётся фискальному регистратору целиком, без детализации продажи, денег и т.п.

10.2.3 Чек по шаблону

Наиболее универсальный режим, позволяет прикладному скрипту вообще не вдаваться в подробности форматирования чека и ограничиться предоставлением объекту `CashMatic.Print.Receipt` информации о выполненном платеже. Скрипт должен предоставить всю возможную информацию, использование или неиспользование её в чеке будет определяться выбранным шаблоном печати (подробности см. Шаблоны печати).

Пример:

```
/*
	Печать чека продажи на 2 позиции:
		- платёж за услугу на 100 руб.
		- 5 конфет по 5 рублей.
	при условии внесения наличных 150 руб.
*/
CashMatic.Print.Reset();	//начальная установка объекта печати
//CashMatic.Print.Fiscal=false; //если раскомментировать эту строку,
			//	то даже на фискальном регистраторе чек
			//	будет печататься как на обычном принтере,
			//	без фискализации.
CashMatic.Print.Receipt.Template="Торговый.txt"; //выбираем шаблон
CashMatic.Print.Receipt.Title = "Пример"; //заголовок чека (отображается
			//	только если в шаблоне есть ссылка)
CashMatic.Session.AddProp("ТелефонСлужбыПоддержки", "00000000");
			//	дополнительная информация, на которую может
			//	ссылаться шаблон печати
CashMatic.Print.Receipt.Id = "DEMO1234"; //идентификатор чека
			//(отображается только если в шаблоне есть ссылка)
CashMatic.Print.Receipt.Code = "200000000001"; //код платёжного документа
			//(отображается только если в шаблоне есть ссылка)

//Добавляем позицию в список продажи:
Item = CashMatic.Print.Receipt.SaleItems.Add();
Item.Name = "Платёж за услугу";		//название позиции в списке продажи
Item.Price = 100;	//цена товарной единицы
Item.Quantity = 1;	//количество оплаченного товара
Item.TextBelow = "Спасибо за своевременную оплату";	//строка печати
			//	ниже товарной позиции

//Добавляем позицию в список продажи:
Item = CashMatic.Print.Receipt.SaleItems.Add();
Item.Name = "Конфета"; //название позиции в списке продажи
Item.Price = 5;	//цена товарной единицы
Item.Quantity = 5;	//количество оплаченного товара

//количество внесённых наличных:
CashMatic.Print.Receipt.Cash = 150;

//собственно печать чека:
if(CashMatic.Print.PrintOut())
{	//при успешной печати:
	SendInfoToServer(
		CashMatic.Print.Receipt.Terminal,
		CashMatic.Print.Receipt.Date,
		CashMatic.Print.Receipt.Time,
		CashMatic.Print.Receipt.Id,
		CashMatic.Print.Receipt.Total,
		CashMatic.Print.Receipt.Cash
	);//предполагаемая функция прикладной программы,
	//	отсылающая на сервер данные чека: имя терминала,
	//	дата, время, итоговая сумма, внесённые наличные
	//	- всё в том виде, как было напечатано на чеке.
	//(функция SendInfoToServer не написана, указана
	//	только для иллюстрации)
}
```

Свойства `CashMatic.Print.Receipt.Id` (идентификатор чека) и `CashMatic.Print.Receipt.Code` (код чека) - это просто строки, которые будут напечатаны в соот-

ветствующих местах в чеке, их значение и смысл задаются разработчиками прикладного скрипта и шаблона чека.
Как правило, на чеке нужно отдельно показать номер чека и номер лицевого счёта, по которому выполнен платёж.
Кроме того, на фискальном регистраторе в чеке может быть напечатан штрих-код, который должен однозначно идентифицировать чек.
Предполагаемые (но не обязательные) назначения для этих свойств:

- `Id` - человекочитаемый идентификатор чека;
- `Code` - штрих-код чека.

Шаблон печати может ссылаться не только на свойства объекта `CashMatic.Print.Receipt` и т.п., но и на переменные платёжного сеанса.

Обратите внимание, итоговая стоимость чека составляет 125 руб., внесено 150 руб., при фискальной печати на чеке будет показана сдача 25 руб. и сделана соответствующая запись в фискальной памяти. При печати на обычном принтере язык описания шаблонов позволяет также указать сдачу, и даже долг (если внесённая сумма меньше стоимости чека). Чеки со сдачей и долгом имеют смысл, если платёжная система допускает последующие дополнительные расчёты с плательщиком. (Чеки с долгом не могут быть оформлены в фискальном режиме, поскольку этому будет противодействовать фискальный регистратор)

Дополнительно можно задать значения свойствам `TextAbove` и/или `TextBelow` (текст над/текст под) для каждого элемента (позиции) в списке продажи.
Нельзя задать значения свойствам `TextAbove` и/или `TextBelow` (текст над/текст под) всего списка продажи (`CashMatic.Print.Receipt.SaleItems.TextAbove` / `CashMatic.Print.Receipt.SaleItems.TextBelow`) - это поддерживается только в режиме фискального чека без шаблона.

Значения свойств подставляются в выходной текст вместо соответствующих макроимён исходного текста шаблона печати.
Значения, задаваемые произвольным текстом (например, `CashMatic.Print.Receipt.Title = "Пример"`), подставляются как есть, без дополнительной макрообработки их текста или текста, полученного в результате их подстановки. Атрибуты форматирования должны задаваться шаблоном и не могут задаваться внутри значений полей.

Следует обратить внимание, что на выводимый текст действуют параметры печати, заданные в Панели управления **NDemia CashMatic** - преобразование кодировки и обработка %-кодов форматирования (на фискальном регистраторе эти операции не поддерживаются, независимо от настройки параметров). Если использование кодов форматирования выключено, то атрибуты шрифта всё равно удаляются из текста, а не пропускаются на печать (атрибуты шрифта рассматриваются как элементы языка описания шаблонов печати, в отличие от режима "текст без шаблона").
Если Вы организуете печать через `CashMatic.Print.Receipt`, то Вы не должны изменять значение `CashMatic.Print.Text`, иначе печать автоматически переключится в режим "текст без шаблона", без фискализации.

Примерный вид чека, который может быть напечатан этим скриптом

```
NDemia CashMatic: Пример
4 октября 2011 г. 18:40:16
Код платежа: DEMO1234
Платёж за услугу    100,00    *1,00
Конфета   5,00 *5,00
Итого 125,00 руб.
Оплачено 150,00 руб. через терминал CASHMATIC
```

Обратите внимание: вид чека в существенной степени зависит от дизайна шаблона, скрипт только предоставляет данные, заполнение полей делается по выбранному шаблону.

Такой же скрипт, шаблон "Простой"

```
NDemia CashMatic: Пример
04.10.2011 18:39 CASHMATIC
Код платежа: DEMO1234
Сумма:       150,00 руб.
```

Такой же скрипт, шаблон "Развёрнутый"

```
NDemia CashMatic: Пример
04.10.2011 18:40
Код платежа: DEMO1234
1. Платёж за услугу 100,00    *1,00     = 100,00
Спасибо за своевременную оплату
2. Конфета     5,00 *5,00     = 25,00

Итого 125,00 руб.
Оплачено наличными 150,00 руб. через терминал
CASHMATIC
Покупателю сдача 25,00 руб.
Телефон службы поддержки: 00000000

Код документа: 200000000001
```

Другие примеры подготовки данных для печати чека по шаблону:

- см. Примеры шаблонов печати из комплектации программного продукта;
- см. Пример платёжного интерфейса (`finish.html`).

10.2.4 Форматирование шрифта

Форматирование шрифта символов зависит от возможностей конкретного оборудования, используемого для печати чеков. Однако разработчик дизайна чека может абстрагироваться от конкретики оборудования и пользоваться средствами описания форматирования **NDemia CashMatic**. Какие опции форматирования будут в действительности применяться при печати на бумаге - это будет определяться настройками конкретного терминала.
Однако определённо можно сказать, что форматирование шрифта символов для принтера и для фискального регистратора различается, поскольку фискальный регистратор не позволяет изменять шрифт отдельных символов, но разрешает указать шрифт строки целиком как номер по таблице шрифтов, прошитой в оборудовании.

При использовании шаблона форматирование шрифта символов задаётся самим шаблоном.
Если шаблон не используется, то атрибуты шрифта для принтера управляются кодами форматирования (подмножество языка описания шаблонов печати **NDemia CashMatic**).
Поддерживаются следующие атрибуты:

Код	Формат	Пример
%%	%	*Для вывода знака "процент" символ должен быть удвоен.*
%b	жирный шрифт	`Выделенное слово будет напечатано %b`**`жирным`**`%b шрифтом.`
%i	наклонный шрифт (курсив)	`Выделенное слово будет напечатано %i`*`наклонным`*`%i шрифтом.`
%r	инверсия	`Выделенные слова будут напечатаны %r`белым по чёрному`%r.`
%u	подчёркивание	`Выделенное слово будет %u`подчёркнуто`%u при печати.`

Обратите внимание, обработка кодов форматирования зависит от настройки, заданной в Панели управления **NDemia CashMatic** ("использовать коды форматирования"). Если использование кодов форматирования выключено, то все перечисленные %-коды при использовании шаблона будут удаляться из выводимого текста, а без шаблона - пропускаться на печать. Если включено - управляющие комбинации будут извлекаться из выводимого текста, даже если соответствующее форматирование не поддерживается оборудованием (например, на фискальном регистраторе будут удаляться все %-коды)

Выбор шрифта строки для фискального регистратора выполняется вставкой в текст строки дополнительной управляющей команды: ESC <номер> (**`hex: 1B n`**, где n: hex 30..39) - строка должна начинаться с этой последовательности (данная команда обрабатывается приложением **NDemia CashMatic** при выводе текста и не является командой самого фискального регистратора). При использовании шаблона следует вместо команды использовать макроимя `%(шрифт)` - в этом случае управляющий код будет вставляться только при выводе на фискальный регистратор и не будет вставляться при выводе на принтер.

10.3 Шаблоны печати

Шаблоны печати применяются для организации вида выходных документов (чеков, квитанций и т.п. - обобщённо выходные печатные документы называются чеками). Прикладная программа (java-скрипт внутри HTML-страницы) передаёт объектам печати всю информацию о принятом платеже и полностью освобождается от необходимости управлять составом, расположением и форматом полей на чеке. Всё доступное прикладной программе управление в этом случае сводится к выбору шаблона печати (см. Чек по шаблону).

Шаблоны печати (или шаблоны чеков) представляют собой заготовки текста, в который вставлены некоторые специальные указания по разметке выходного документа.
Непосредственно перед печатью выполняется программное преобразование шаблона в конечный выходной текст для печати. Одни фрагменты текста шаблона удаляются из выходного текста, другие - заменяются на программно вычисляемые текстовые значения, некоторые фрагменты могут быть размножены и вставлены несколько раз, с разными подстановками значений. В целом процесс вычисления конечного текста путём различных программных преобразований текста шаблона называется макрообработкой шаблона.
Макрообработка выполняется над особо размеченными фрагментами текста шаблона - макроименами и макроблоками.
Макроимена - это некоторые специальные комбинации знаков, которые при печати заменяются программно вычисляемыми значениями (эта замена называется макроподстановкой). Например, `"%t"` заменяется на текущее время, `"%(cash)"` - на принятую сумму наличных денег, и т.д. Разработчик шаблона печати может использовать целый ряд предопределённых (заранее предусмотренных) макроимён, а также может вставлять в шаблон ссылки на переменные сеанса приёма платежа.
Макроблоки - это целые куски текста шаблона, которые обрабатываются в зависимости от типа макроблока - либо удаляются из выходного текста, либо остаются в выходном тексте, либо копируются и вставляются несколько раз (например, чек продажи может содержать несколько товарных позиций, с разными наименованиями и ценами, блок продажи описывается в обобщённом виде для одной товарной позиции, а вставляется в выходной текст в нескольких экземплярах, по одному для каждой товарной позиции, каждый раз с соответствующими макроподстановками наименования и цены товара).
Макроблоки могут содержать внутри себя другие (вложенные) макроблоки. Если макроблок удаляется из выходного текста, то и все вложенные макроблоки удаляются без обработки. Иначе, если макроблок остаётся в выходном тексте (или вставляется дополнительный экземпляр, построенный на основании макроблока), то все вложенные в него макроблоки обрабатываются в соответствии с их типами.
Макроимена и типы макроблоков могут иметь несколько альтернативных имён. Например, сумма принятых наличных денег может быть записана в шаблоне чека как `"%m"`, `"%(нал)"`, `"%(nal)"` или `"%(cash)"` - разработчик может выбирать любую форму по своему усмотрению. Аналогично, макроблок позиции продажи может называться `"прод"`, `"prod"` и `"sale"` - можно использовать любой вариант, как будет удобнее.
Шаблоны печати могут содержать знаки форматирования шрифта. Для фискального регистратора поддерживается только выбор шрифта строки целиком (а также режим штрих-кода для строки целиком) - для этого разработчик шаблона должен вставить соответствующую команду как макроимя в первой позиции выходной строки.

Шаблоны печати создаются и загружаются как текстовые файлы, поддерживаются следующие форматы:

- простой 8-битный текстовый формат, кодировка русского языка: Windows cp1251;
- Юникод, преобразованный UTF8.

> **Совместимость:** Юникод (UTF-8) в шаблонах печати поддерживается в **NDemia CashMatic**, начиная с версии **2.7.0**

Файлы шаблонов содержат разметку документов и полей на языке описания шаблонов печати **NDemia CashMatic**.
По умолчанию шаблоны чеков располагаются в папке `PrintTemplates` внутри папки установки программного продукта **NDemia CashMatic** (**`"\Program Files\NDemia\CashMatic"`**). При установке программы эта папка создаётся и в ней размещаются примеры шаблонов (все примеры полнофункциональны и могут использоваться без модификаций).

10.3.1 Язык описания шаблонов печати

см. также: Шаблоны печати | Примеры шаблонов | Чек по шаблону

Язык описания шаблонов печати **NDemia CashMatic** является макроязыком разметки документа.
При получении команды печати (вызов `CashMatic.Print.PrintOut()`), если задано имя шаблона (`CashMatic.Print.Receipt.Template`), открывается соответствующий текстовый файл и выполняется его макрообработка на основе языка, описанного в разделе Синтаксис. Текст, полученный в результате макрообработки, и является текстом выходного документа. В случае печати фискальных чеков фискальный регистратор формирует ряд строк самостоятельно, на основании информации, получаемой от прикладной программы.

Синтаксическими единицами языка описания шаблонов являются макроимена и макроблоки (см. соответствующие разделы).
Прикладной скрипт может передавать данные в текст печати через соответствующие поля шаблона - см. раздел Макроподстановка значений переменных платёжного сеанса.
Дополнительные сведения по управлению шрифтами и отображению специальных типов значений (дата, время, деньги) приведены в разделе Форматирование.

10.3.1.1 Синтаксис

см. также: Шаблоны печати | Примеры шаблонов | Объекты печати

Текст шаблона печати при его макрообработке рассматривается на двух уровнях: как поток символов и как многоуровневая иерархия блоков (макроблоков). Исходя из этого, синтаксические правила языка описания шаблонов печати **NDemia CashMatic** разделены на две группы:

- Общие синтаксические правила;
- Синтаксические правила для макроблоков.

10.3.1.1.1 Общие синтаксические правила

см. также: Шаблоны печати Синтаксис Макроимена

Общие синтаксические правила языка описания шаблонов печати **NDemia CashMatic**.

- все символы, не требующие макрообработки, выводятся на печать.
- % (знак процента) называется мета-символом, если он встречается в тексте шаблона, то дальнейшая обработка текста зависит от непосредственно следующего за мета-символом знака.
- если за мета-символом следует знак процента ("%%") или непосредственно следует конец текста, то в выходной текст вставляется один знак процента.
- если за мета-символом следует левая круглая скобка ("% ("), то это - макроимя. Собственно макроименем является текст от "% (" до правой круглой скобки (") ") включительно (или до конца текста шаблона). Макроимя при обработке заменяется на определённый соответствующий ему текст, вычисленный программно. Верхний/нижний регистр букв в макроименах не различается. Неизвестные макроимена просто удаляются (т.е. их значения считаются пустыми).
- если за мета-символом следует левая фигурная скобка ("%{"), то это - макроблок. Конец макроблока - комбинация "%}" (обратите внимание, процент перед скобкой, не "}%") или конец текста шаблона. Макроблоки могут вкладываться друг в друга (по обычным правилам вложения скобочных конструкций). Обработка макроблока зависит от типа макроблока - см. Макроблоки.
- если за мета-символом следует любой другой знак, то эта пара знаков (процент и другой знак) может рассматриваться как короткое макроимя. Если это макроимя имеет определённое значение, то в выходной текст вставляется это значение, в противном случае данная пара знаков пропускается в выходной текст без обработки (не удаляется). Однако разработчик шаблона не должен на это полагаться, поскольку в будущих версиях могут быть определены новые комбинации знаков с какой-либо обработкой. При необходимости вывода на печать знака "%" его следует удваивать ("%%").
- обрабатывается только исходный текст, за один проход. Текст, полученный в результате макроподстановок, дополнительно не обрабатывается.
- все неуказанные символы, включая разделители (пробелы, переводы строк и т.п.) являются значащими (входят в обрабатываемый текст), поэтому дополнительные разделители (для улучшения читаемости текста шаблона) можно вставлять только в комментариях.

10.3.1.1.2 Синтаксические правила для макроблоков

см. также: Шаблоны печати Синтаксис Макроблоки

Язык описания шаблонов печати **NDemia CashMatic**, синтаксические правила для макроблоков:

- непосредственно за началом макроблока ("%{") следует заголовок макроблока, заголовок начинается с имени типа макроблока. Именем типа считается весь текст от первого знака после "%{" либо до первого двоеточия (":", невключительно), либо до левой круглой скобки ("("), либо до конца текста шаблона. Выполняемая обработка макроблока зависит от его типа.
- если непосредственно за именем типа макроблока следует левая круглая скобка ("("), то далее весь текст до правой круглой скобки (")") считается параметром макроблока. Параметр макроблока может содержать любые символы, включая мета-символы, двоеточия и т.п., концом параметра является только правая круглая скобка (")"), возможность вложений скобок поддерживается (т.е. общее число открывающих скобок в параметре должно соответствовать числу закрывающих). Непосредственно за правой круглой скобкой (")") должно следовать двоеточие (":"). Выполняемая обработка текста параметра макроблока зависит от типа макроблока.
- если тип блока неизвестен, то блок полностью включается в выходной текст как есть, включая скобочные комбинации ("%{" и "%}") и весь текст между ними. Если в блоке есть вложенные блоки, то к ним применяются обычные правила обработки, в зависимости от их типов. Разработчик шаблонов не должен полагаться на порядок обработки блоков неизвестного типа, поскольку в будущих версиях могут быть введены новые синтаксические правила.
- текстом макроблока считается текст, начинающийся с первого знака после двоеточия, закрывающего тип макроблока, до последнего знака, предшествующего концу макроблока (закрывающей комбинации "%}" или концу текста шаблона).
- в результате обработки текст макроблока включается в выходной текст от 0 до N раз (в зависимости от типа макроблока, параметра или параметров макроблока, и других условий). При многократном включении (циклический макроблок) текст включается каждый раз последовательно (первый символ i+1-го включения следует за последним символом i-того включения).
- внутри макроблока могут меняться набор и значения действующих макроимён и типов макроблоков. При многократном включении (циклический макроблок) набор и значения действующих макроимён и типов макроблоков могут меняться в каждом проходе.
- внутри макроблока могут быть вложены другие макроблоки. Обработка вложенных макроблоков выполняется столько раз, сколько раз включается в выходной текст охватывающий макроблок (от 0 до N).
- конец строки включается в текст макроблока как символ. Конец макроблока не является концом строки, если текст макроблока должен включать в себя строку целиком или несколько целых строк, то конец макроблока должен находиться на следующей строке шаблона.
- макроблок с пустым типом (начинающийся с комбинации "%{:") вставляется в выходной текст один раз, с удалением начальной комбинации "%{:", конечной комбинации "%}" и обработкой содержащихся внутри макроимён и макроблоков (это положение гарантируется для будущих версий языка описания шаблонов, блоки с пустым типом могут использоваться для дополнительной разметки шаблонов в любых целях, определяемых разработчиками шаблонов и прикладных программ).

10.3.1.2 Макроимена

см. также: Шаблоны печати Объекты печати Макроблоки

Макроимена языка описания шаблонов печати **NDemia CashMatic** делятся на две группы:

- полные макроимена - вида "знак процента-левая круглая скобка-имя-правая круглая скобка";
- короткие макроимена - вида "знак процента-буква".

10.3.1.2.1 Полные макроимена

см. также: Шаблоны печати Макроблоки Короткие макроимена

Макроимя	Альтернативные имена (•)	Область действия (••)	Обработка	Примечание
%(время)	%(time), %(vremya)	*везде*	время на момент печати чека, с секундами	для времени без секунд можно использовать короткую форму: **%t**; см. свойство **CashMatic.Print.Receipt.Time**; см. Форматирование данных
%(дата)	%(data), %(date)	*везде*	дата на момент печати чека, длинный формат даты	для краткой даты можно использовать короткую форму: **%d**; см. свойство **CashMatic.Print.Receipt.Date**; см. Форматирование данных
%(долг)	%(debt), %(dolg)	*везде*	долг как **%(итого)** минус **%(нал)** (только если **%(итого)** > **%(нал)**, иначе ноль)	см. свойство **CashMatic.Print.Receipt.Debt**; см. Форматирование данных
%(загол)	%(title), %(zagol)	*везде*	заголовок чека (•••)	задаётся через свойство **CashMatic.Print.Receipt.Title**; можно использовать короткую форму: **%c**
%(ид)	%(id)	*везде*	идентификатор/номер чека (•••)	задаётся через свойство **CashMatic.Print.Receipt.Id**; можно использовать короткую форму: **%n**
%(итого)	%(itogo), %(total)	*везде*	итоговая стоимость чека, как сумма стоимостей всех позиций списка продажи	см. свойство **CashMatic.Print.Receipt.Total**; см. Форматирование данных
%(код)	%(kod), %(code)	*везде*	код документа (чека) (•••)	задаётся через свойство **CashMatic.Print.Receipt.Code**; можно использовать короткую форму: **%o**
%(колво)	%(kolvo), %(quantity)	*позиция списка продажи*	количество товарных единиц по позиции продажи	задаётся через свойство **CashMaticSaleItem.Quantity**; см. Форматирование данных
%(назв)	%(name), %(nazv)	*позиция списка*	название позиции продажи (•••)	задаётся через свойство **CashMaticSaleItem.Name**

		продажи		
%(нал)	%(cash), %(nal)	*везде*	сумма внесённых наличных денег	можно использовать короткую форму: **%m**; см. свойство **CashMatic.Print.Receipt.Cash**; см. Форматирование данных
%(номер)	%(nomer), %(number)	*позиция списка продажи*	порядковый номер позиции	позиции нумеруются автоматически, в порядке добавления в список продажи (см. **CashMatic.Print.Receipt.SaleItems**)
% (перем:Имя)	%(prop:**Имя**), %(var:**Имя**)	*везде*	значение переменной платёжного сеанса (здесь **Имя** - имя переменной)	значение переменной задаётся через вызов **CashMatic.Session.AddProp(Имя, Значение)**. см. Сеанс приёма платежа см. Макроподстановка значений переменных платёжного сеанса **Совместимость:** поддерживается, начиная с версии **2.6.2**
%(сдача)	%(change), %(sdacha)	*везде*	сдача как **%(нал)** минус **%(итого)** (только если **%(нал)** > **%(итого)**, иначе ноль)	см. свойство **CashMatic.Print.Receipt.Change**; см. Форматирование данных
%(стоим)	%(cost), %(stoim)	*позиция списка продажи*	стоимость позиции как **%(цена)** * **%(колво)**	см. Форматирование данных
%(текств)	%(tekstv), %(texta)	*позиция списка продажи*	текст над строкой продажи (•••)	задаётся через свойство **CashMaticSaleItem.TextAbove**
%(текстн)	%(tekstn), %(textb)	*позиция списка продажи*	текст под строкой продажи (•••)	задаётся через свойство **CashMaticSaleItem.TextBelow**
%(терминал)	%(terminal)	*везде*	имя терминала на момент печати чека (•••)	см. свойство **CashMatic.Print.Receipt.Terminal**
%(цена)	%(cena), %(price)	*позиция списка продажи*	цена товарной единицы по позиции	задаётся через свойство **CashMaticSaleItem.Price**; см. Форматирование данных
%(шрифтN)	%(fontN)	*везде*	выбор шрифта для строки на фискальном регистраторе	где N: 0..9; см. Выбор шрифта
%(штрихКод)	%(barCode)	*везде*	выбор режима печати штрих-кода для строки на фискальном регистраторе	только выбор режима, значение штрих-кода должно быть определено дополнительно; см. Печать штрих-кода *Рекомендуется для указания штрих-кода печатаемого документа использовать свойство **CashMaticSaleItem.Code**.* **Совместимость:** поддерживается, начиная с версии **2.6.0**

(•) любое из имён может использоваться равноправно, по усмотрению разработчика шаблона
(••) Области действия макроимён:

- "везде" означает, что имя действительно в любом месте шаблона, кроме исключённых блоков.

- "позиция списка продажи" означает, что имя действительно только внутри блока продажи (макроблок "`прод`"). Блок продажи является циклическим макроблоком, и вставляется в выходной текст столько раз, сколько элементов типа `CashMaticSaleItem` добавлено в коллекцию `CashMatic.Print.Receipt.SaleItems`. Макроимена с областью действия "позиция списка продажи" определяются отдельно для каждой вставки макроблока "`прод`" и недействительны вне блока "`прод`".

(•••) в значениях полей, содержащих произвольный текст, знаки "`%`" не обрабатываются, таким образом, макроимена и атрибуты форматирования не учитываются, удвоение знака процента не требуется. Например, если терминал имеет имя "`%d`", то макроимя `%(терминал)` при печати будет заменено на текст "`%d`", а не на текущую дату.

10.3.1.2.2 Короткие макроимена

см. также: Шаблоны печати Макроимена Атрибуты шрифта

К коротким макроименам языка описания шаблонов печати **NDemia CashMatic** (комбинации %-<буква>) относятся команды форматирования (см. Атрибуты шрифта) и сокращённые эквиваленты некоторых полных макроимён (при использовании коротких макроимён в некоторых случаях применяется более короткий текстовый формат значений).

Комбинация	Мнемоника (•)	Обработка	Примечание
%b	Bold	пропускается на уровень форматирования символов	включение/выключение жирного шрифта (см. Атрибуты шрифта)
%c	Caption	заголовок чека	эквивалент **%(загол)**
%d	Date	дата на момент печати чека	почти эквивалент **%(дата)** - используется короткий формат даты (см. Форматирование данных)
%i	Italic	пропускается на уровень форматирования символов	включение/выключение наклонного шрифта (курсива) (см. Атрибуты шрифта)
%m	Money	сумма принятых наличных денег	эквивалент **%(нал)**
%n	Name	идентификатор чека	эквивалент **%(ид)**
%o	cOde	код документа (чека)	эквивалент **%(код)** **Совместимость:** поддерживается, начиная с версии **2.6.0**
%r	inveRt	пропускается на уровень форматирования символов	включение/выключение инверсной печати (см. Атрибуты шрифта)
%t	Time	время на момент печати чека	почти эквивалент **%(время)** - выводится время без секунд (см. Форматирование данных)
%u	Underline	пропускается на уровень форматирования символов	включение/выключение подчёркивания (см. Атрибуты шрифта)

(•) мнемоника как таковая нигде не используется, приводится только для пояснения выбора буквы в комбинации.

10.3.1.3 Макроблоки

см. также: Шаблоны печати Синтаксис Макроимена

Тип макроблока	Альтернативные имена (•)	Значение	Обработка (••)	Примечание
коммент	comment, rem, прим, prim	комментарий, примечание	никогда не включается	-
если	if, esli	если определено непустое значение параметра	включается, если задан параметр **И** если параметр является макроименем **И** если макроимя имеет непустое значение. (•••)	см. Макроимена **Совместимость:** поддерживается, начиная с версии **2.6.2** **Совместимость:** альтернативное имя **"esli"** поддерживается, начиная с версии **2.7.1**
нал	cash, nal	если есть наличные	включается при условии внесения некоторого ненулевого количества наличных денег в ходе приёма платежа	см. свойство **CashMatic.Print.Receipt.Cash**; см. макроимена **%(нал)**, **%m**; см. макроблок "**ненал**"
ненал	nocash, нетнал, nenal, netnal	если нет наличных	включается в при условии невнесения никаких наличных денег в ходе приёма платежа	см. свойство **CashMatic.Print.Receipt.Cash**; см. макроимена **%(нал)**, **%m**; см. макроблок "**нал**"
сдача	change, сд, sd, sdacha	если есть сдача	включается при условии внесения в ходе приёма платежа суммы наличных, превышающей стоимость чека	см. свойство **CashMatic.Print.Receipt.Change**; см. макроимя **%(сдача)**; см. макроблок "**безсдачи**"
безсдачи	nochange, бсд, bsd, bezsdachi	если нет сдачи	включается при условии внесения в ходе приёма платежа суммы наличных, меньшей или равной стоимости чека	см. свойство **CashMatic.Print.Receipt.Change**; см. макроимя **%(сдача)**; см. макроблок "**сдача**"
долг	debt, dolg	если есть долг	включается при условии внесения в ходе приёма платежа суммы наличных, меньшей стоимости чека	см. свойство **CashMatic.Print.Receipt.Debt**; см. макроимя **%(долг)**; см. макроблок "**бездолга**"
бездолга	nodebt, бд, bd, bezdolga	если нет долга	включается при условии внесения в ходе приёма платежа суммы наличных, большей или равной стоимости чека	см. свойство **CashMatic.Print.Receipt.Debt**; см. макроимя **%(долг)**; см. макроблок "**долг**"
фискал	fiscal, фр, fiskal, fr	если фискальный режим	включается при условии печати фискального чека на соответствующем оборудовании	см. свойство **CashMatic.Print.Receipt.Fiscal**; см. макроблок "**нефискал**"

нефискал	nonfiscal, нфр, nefiskal, nfr	`если не фискальный режим`	включается при условии печати нефискального чека	см. свойство **CashMatic.Print.Receipt.Fiscal**; см. макроблок "**фискал**"
прод	sale, prod	`продажа`	в нефискальном режиме: циклический макроблок, включается для каждой позиции списка продажи; в фискальном режиме этот блок полностью исключается и на бумаге заменяется чеком продажи, формируемым фискальным регистратором (если данный блок отсутствует в шаблоне, то чек продажи печатается после текста шаблона; если данный блок присутствует в шаблоне несколько раз, то чек продажи печатается на месте последнего включения блока).	см. свойство **CashMatic.Print.Receipt.SaleItems**; см. Синтаксические правила для макроблоков; см. Макроимена

(•) любое из имён может использоваться равноправно, по усмотрению разработчика шаблона
(••) в графе Обработка под включением блока подразумевается включение текста макроблока в выходной текст с обработкой всех вложенных блоков.
(•••) Обратите внимание: условию макроблока `"если"` удовлетворяет только непустое значение - строка, содержащая более нуля знаков, любых, в том числе пробелов. Арифметические/логические операции не выполняются, числа и логические значения рассматриваются только как текст.

- Пример 1: `%{если(%(код)):Текст%}` - **Текст** включается в выходной текст, если задан непустой код документа (см. макроимя `%(код)`)
- Пример 2: `%{если(%(перем:Имя)):Текст%}` - **Текст** включается в выходной текст, если в платёжном сеансе переменная `Имя` задана и имеет непустое значение.
- Пример 3: `%{если:Текст%}` - параметр не задан, поэтому условие всегда ложно, **Текст** никогда не включается в выходной текст.

10.3.1.4 Макроподстановка значений переменных платёжного сеанса

см. также: Шаблоны печати Платёжный сеанс Макроимена

Разработчик прикладных скриптов и компонентов расширений имеет возможность создавать собственные значения для подстановки в текст печати по шаблону. Для этого в шаблон печати вставляются специальные макроимена, которые при подготовке текста печати автоматически заменяются значениями переменных платёжного сеанса.
Для макроимени вида «**перем:Имя**» (или «**var:Имя**», или «**prop:Имя**») при печати выполняется подстановка значения переменной «**Имя**».
Может использоваться любой из префиксов «**перем:**», «**var:**», или «**prop:**», по усмотрению разработчика шаблона. Двоеточие обязательно, непосредственно после двоеточия без дополнительных разделителей указывается имя переменной платёжного сеанса.

Допустим, наш прикладной скрипт при приёме платежа требует указывать некоторый номер лицевого счёта и фамилию плательщика, и перед нами стоит задача указать эти данные в чеке, формируемом по шаблону.
Нужные нам значения не предусмотрены в свойствах чека и макроименах шаблонов печати, но мы можем добавить собственные именованные значения.
Для этого в ходе платёжного сеанса мы должны создать соответствующие переменные, например:

```
CashMatic.Session.AddProp("ЛицевойСчёт", 1234);
CashMatic.Session.AddProp("Фамилия", «Иванов»);
```

А в шаблон чека вставить ссылки на эти переменные, например:

```
%{rem:
        Пример шаблона чека, в котором используется
макроподстановка значений переменных платёжного сеанса
%}
Лицевой счёт: %(перем:ЛицевойСчёт)
Плательщик:   %(перем:Фамилия)
Оплачено:     %m руб.
%d %t %(терминал)
%{rem:

        ЛицевойСчёт и Фамилия — наши собственные значения, которые при
печати будут подставляться из CashMatic.Session
        Что такое %m, %d, %t и %(терминал) — см. Макроимена
%}
```

Примерный вид чека, который может быть напечатан по такому шаблону

```
Лицевой счёт: 1234
Плательщик:   Иванов
Оплачено:     200 руб.
02.10.2011 19:13 CASHMATIC
```

Совместимость: макроподстановка переменных платёжного сеанса в шаблонах печати поддерживается в **NDemia CashMatic**, начиная с версии **2.6.2**

10.3.1.5 Форматирование

см. также: Шаблоны печати | Короткие макроимена | Форматирование шрифта

В этом разделе содержатся сведения о возможностях управления форматом шрифта символов при печати на принтере (см. Атрибуты шрифта) и при печати на фискальном регистраторе (см. Выбор шрифта строки, Печать штрих-кода), а также разъясняется форматирование вывода специальных типов данных (дата, время, денежные суммы) (см. Форматирование данных).

10.3.1.5.1 Атрибуты шрифта

Язык описания шаблонов печати **NDemia CashMatic** предоставляет следующие возможности по управлению атрибутами шрифта печати:

- печать жирным шрифтом (`%b`);
- наклон (курсив) (`%i`);
- инверсия ("негативная" печать, белым по чёрному) (`%r`);
- подчёркивание (`%u`).

Атрибуты управляются комбинациями знака процента и латинской буквы. Такие комбинации действуют как "триггерные скобки" - каждое нечётное вхождение включает соответствующий атрибут, и каждое чётное выключает.
Например, если шаблон содержит фразу:

```
Печать %iнаклонным%i шрифтом
```

(и это первое и второе употребление `%i` от начала шаблона), то слово "наклонным" будет напечатано курсивом.

Печать с атрибутами форматирования символов требует соответствующей поддержки оборудованием (т.е. принтер должен "уметь" выполнять указанные изменения шрифта), а также требуется правильная настройка системы команд принтера (см. вкладку Принтер в Панели управления **NDemia CashMatic**).
Фискальный регистратор управление форматом символов не поддерживает (однако у него есть возможность выбора шрифта для целой строки).

Если оборудование печати не поддерживает какие-либо атрибуты форматирования (или если использование кодов форматирования выключено в параметрах принтера в Панели управления **NDemia CashMatic**), комбинации "процент-буква" будут просто удаляться из выходного текста.

Коды форматирования (комбинации "процент-буква") обрабатываются только по исходному тексту шаблона и не обрабатываются внутри подставляемых значений. Таким образом, нельзя задать форматирование в самом подставляемом тексте какого-либо поля.
Коды форматирования могут использоваться также и при печати без шаблонов (см. Текст без шаблона).

10.3.1.5.2 Выбор шрифта строки

Выбор шрифта строки поддерживается только на фискальном регистраторе.
Шрифт выбирается по номеру из таблицы шрифтов, прошитой в оборудовании.
Для указания нужного шрифта в первой позиции печатной строки должно быть вставлено макроимя `%(шрифтN)` (или `%(fontN)`), где N - число от 0 до 9 (на самом деле поддерживаемых шрифтов будет скорее всего меньше - определяется конкретной моделью оборудования). Кроме указанных макроимён может использоваться `%(штрихКод)` (или `%(barCode)`) - см. Печать штрих-кода (режим печати штрих-кода определяется как разновидность шрифта строки).
Макроимя может использоваться только в исходном тексте шаблона, при его вставке в строчное значение какого-либо подставляемого поля шаблона оно будет просто напечатано и не будет иметь никакого эффекта в плане форматирования.
Если печать будет выполняться на принтере (не на фискальном регистраторе), то указание номера шрифта ни на что не повлияет.
Если выводимая строка будет слишком длинной и при печати будет разбита на несколько строк, указан-

ный в первой позиции шрифт будет применён ко всем частям исходной строки.

Определённую сложность для разработчика шаблона может представлять правильное определение первой печатной позиции строки - строка шаблона (в файле) может начинаться с каких-либо синтаксических конструкций, которые в выходном тексте должны быть удалены.

Пример правильного определения первой печатной позиции:
```
%{коммент: Комментарий
%}%(шрифт1)Текст, который должен быть напечатан шрифтом №1
```
в выходном тексте фрагмент **"%{коммент: Комментарий...%}"** будет удалён, поэтому `%(шрифт1)` находится действительно на первой печатной позиции строки.

Пример неправильного определения первой печатной позиции:
```
%{коммент: Комментарий
%}%(шрифт1)Текст, который должен быть напечатан шрифтом №1
```
в выходном тексте фрагмент **"%{коммент: Комментарий...%}"** не будет удалён (как макроблок с неизвестным типом - из-за ошибки в написании), поэтому `%(шрифт1)` не попадает на первую позицию, и, соответственно, управление шрифтом не срабатывает (см. Формирование текста. Форматирование шрифта.)

Ещё один пример неправильного определения первой печатной позиции:
```
%{сдача:ВЫДАТЬ СДАЧУ %(сдача) руб.%}%(шрифт1)Текст печати шрифтом №1
```
при проверке шаблона `%(шрифт1)` возможно и сработает, но как только возникнет чек со сдачей - произойдёт ошибка форматирования.

10.3.1.5.3 Печать штрих-кода

Печать штрих-кода документа (чека) поддерживается только на фискальном регистраторе.
Режим печати штрих-кода задаётся аналогично штрифту строки (см. Выбор шрифта строки) - в первой позиции печатной строки должно быть вставлено макроимя `%(штрихКод)` (или `%(barCode)`). Макроимя может использоваться только в исходном тексте шаблона, при его вставке в строчное значение какого-либо подставляемого поля шаблона оно будет просто напечатано и не будет иметь никакого эффекта в плане печати штрих-кода.
Если печать будет выполняться на принтере (не на фискальном регистраторе), то указание режима штрих-кода ни на что не повлияет.
Макроимя `%(штрихКод)` задаёт режим печати строки как штрих-кода, но значение штрих-кода - это текст до конца указанной строки. Соответствие длины и значения текста штрих-кода какому-либо стандарту штрих-кодов не проверяется. Рекомендуется программно задавать штрих-код чека через свойство `Code` объекта `CashMaticReceipt`, и, соответственно, подставлять в шаблон чека макроимя `%(код)`.
Типичный фрагмент шаблона чека, задающий печать штрих-кода:
```
%(штрихКод)%(код)
```

Совместимость: печать штрих-кодов поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**

10.3.1.5.4 Форматирование данных

При выводе на печать даты и времени чека используется системный формат, соответствующий настройкам, указанным в Панели управления **Windows**, Язык и региональные стандарты, вкладка Региональные параметры (используются значения настроек, синхронизированные в момент начала платёжного сеанса). При печати фискального чека ряд полей форматируется фискальным регистратором в соответствии с его настройками, в этом случае указания шаблона и системные форматы не действуют.

Дата печати чека выводится через макроимя `%(дата)` (или `%(data)`, `%(date)`). При этом используется системный формат "Полная дата".
Можно вывести дату как макроимя `%d`, в этом случае используется "Краткая дата".

Время печати чека выводится как макроимя `%(время)` (или `%(vremya)`, `%(time)`). Используется системный формат времени, время выводится с секундами.
Можно вывести время как макроимя `%t`, в этом случае печатаются часы и минуты, без секунд.

Денежные значения (`%(цена)`, `%(итого)` и т.п. - полный список см. Макроимена) печатаются по системному формату "Сумма денег" (значения округляются до количества дробных знаков, соответствующего настройке региональных параметров операционной системы, но не более 4).
Числовые значения (например, `%(колво)` - см. Макроимена) печатаются по системному формату "Число" (значения округляются до количества дробных знаков, соответствующего настройке региональных параметров операционной системы).
Дополнительные сведения по округлению данных при вычислениях: см. объекты `CashMaticReceipt` и `CashMaticSaleItem`.

10.3.2 Примеры

При установке программного продукта **NDemia CashMatic KioskBrowser** примеры шаблонов чеков устанавливаются в папке **"\Program Files\NDemia\CashMatic\PrintTemplates"**.
В комплекте три варианта шаблонов:

- "Простой" - по этому шаблону создаются достаточно простые, но вместе с тем информационно полные чеки.
- "Торговый" - показаны основные возможности, предоставляемые языком описания шаблонов.
- "Развёрнутый" - достаточно сложный шаблон, используются практически все возможности языка описания.

Количество шаблонов, их названия и имена файлов никак не регламентируются и могут изменяться по Вашему усмотрению.
Чтобы попробовать все примеры шаблонов в действии, можно воспользоваться примером платёжного интерфейса, поставляемым в комплекте программного продукта - см. Пример.

10.3.2.1 Простой

Пример шаблона при установке программного продукта **NDemia CashMatic KioskBrowser** размещается в файле **"\Program Files\NDemia\CashMatic\PrintTemplates\Простой.TXT"**.
По этому шаблону создаются достаточно простые, но вместе с тем информационно полные чеки.

```
NDemia CashMatic: %c
%d %t %(терминал)
Код платежа: %n
Сумма:       %m руб.
```

(•) Пример из комплектации программного продукта может незначительно отличаться от этого текста.

При печати:

- `%c` заменяется на значение `CashMatic.Print.Receipt.Title`;
- `%d` заменяется на дату печати чека (в формате краткой даты);
- `%t` заменяется на время печати чека (в формате времени, без секунд);
- `%(терминал)` заменяется на имя терминала, заданное в Панели управления **NDemia CashMatic**;
- `%n` заменяется на значение `CashMatic.Print.Receipt.Id`;
- `%m` заменяется на значение `CashMatic.Print.Receipt.Cash` (в формате денежной суммы).

Полный список возможных подставляемых значений: см. Макроимена.
Сведения о форматах данных: см. Форматирование данных.

Для печати чека по этому шаблону прикладная программа (javascript внутри HTML-страницы) должна задать свойства `Title`, `Id` и `Cash` объекта `CashMatic.Print.Receipt`.
Кроме того, нужно указать используемый шаблон и вызвать печать:

```
CashMatic.Print.Receipt.Title="Пример приёма платежа";
CashMatic.Print.Receipt.Id="122";
CashMatic.Print.Receipt.Cash=50;
CashMatic.Print.Receipt.Template="Простой.txt";
CashMatic.Print.PrintOut();    //при ошибке печати возвращает FALSE;
```

Сведения об объекте `CashMatic.Print` и других объектах печати: см. Объекты печати.

Пример чека, напечатанного по шаблону "Простой":

```
NDemia CashMatic: пример приёма платежа
03.10.2011 19:11 CASHMATIC
Код платежа: 122
Сумма:       50,00 руб.
```

10.3.2.2 Торговый

Пример шаблона при установке программного продукта **NDemia CashMatic KioskBrowser** размещается в файле **"\Program Files\NDemia\CashMatic\PrintTemplates\Торговый.TXT"**
В этом примере показаны основные возможности, предоставляемые языком описания шаблонов.

```
NDemia CashMatic: %(загол)
%(дата) %(время)
Код платежа: %(id)
%{прод:%(назв) %(цена) *%(колво)
%}%(шрифт2)%bИтого%b %(итого) руб.
%{нал:Оплачено %(нал) руб. через терминал %(терминал)
%}
```

(•) Пример из комплектации программного продукта может незначительно отличаться от этого текста

При печати:

- `%(загол)` заменяется на `CashMatic.Print.Receipt.Title`;
- `%(дата)` заменяется на дату печати чека (в формате полной даты);
- `%(время)` заменяется на время печати чека (в формате времени, с секундами);
- `%(id)` заменяется на значение `CashMatic.Print.Receipt.Id`;
- блок **`%{прод:...%}`** при нефискальной печати повторяется для каждой позиции продажи, при создании фискального чека фискальный регистратор сам форматирует строки продажи, текст фискального чека вставляется на место блока **`%{прод:...%}`**;
- внутри блока **`%{прод:...%}`** для каждой позиции продажи (элемента списка продажи - см. объект `CashMaticSaleItem`):
  - `%(назв)` - заменяется на `CashMaticSaleItem.Name`;
  - `%(цена)` - заменяется на `CashMaticSaleItem.Price` (в формате денежной суммы);
  - `%(колво)` - заменяется на `CashMaticSaleItem.Quantity` (в формате числа по региональным параметрам операционной системы);
- `%(шрифт2)` - выбор шрифта №2 (по таблице прошитых шрифтов используемого оборудования) действует только на фискальном регистраторе (обратите внимание, указание шрифта работает только в первой позиции выходной строки, хотя в шаблоне эта позиция может быть сдвинута предшествующими синтаксическими конструкциями)
- `%b`**`Итого`**`%b` - выделение жирным шрифтом слова **"Итого"** (не действует на фискальном регистраторе);
- `%(итого)` заменяется на `CashMatic.Print.Receipt.Total`;
- блок **`%{нал:...%}`** (строка **"Оплачено ... руб. через терминал ..."**) печатается только в случае, если были приняты какие-либо наличные (принципиально возможна печать и нулевых чеков, т.е. без приёма наличных);
- `%(нал)` заменяется на `CashMatic.Print.Receipt.Cash` (в формате денежной суммы);
- `%(терминал)` заменяется на имя терминала, заданное в Панели управления **NDemia CashMatic**.

Полный список возможных подставляемых значений: см. Макроимена.
Полный список возможных типов блоков: см. Макроблоки.
Сведения о форматах данных: см. Форматирование данных.
Сведения об управлении шрифтами символов: см. Атрибуты шрифта и Выбор шрифта строки.

Для печати чека по этому шаблону прикладная программа (javascript внутри HTML-страницы) должна задать свойства `Title`, `Id` и `Cash` объекта `CashMatic.Print.Receipt`, в список продажи (объект `CashMatic.Print.Receipt.SaleItems`) нужно добавить хотя бы один элемент (см. объект `CashMaticSaleItem`), для каждого добавленного элемента нужно указать свойства `Name`, `Price` и `Quantity`.
Кроме того, нужно указать используемый шаблон и вызвать печать:

```
CashMatic.Print.Receipt.Title="Пример приёма платежа";
CashMatic.Print.Receipt.Id="112";
CashMatic.Print.Receipt.Cash=50;
var Sale1 = CashMatic.Print.Receipt.SaleItems.Add();
Sale1.Name = "Пример 1";
Sale1.Price = 100;
Sale1.Quantity = 1;
var Sale2 = CashMatic.Print.Receipt.SaleItems.Add();
Sale2.Name = "Пример 2";
Sale2.Price = 500;
CashMatic.Print.Receipt.Template="Торговый.txt";
CashMatic.Print.PrintOut();    //при ошибке печати возвращает FALSE;
```

Сведения об объекте `CashMatic.Print` и других объектах печати: см. Объекты печати.

Пример чека, напечатанного по шаблону "Торговый":

```
NDemia CashMatic: пример приёма платежа
3 октября 2011 г. 19:12:05
Код платежа: 112
Пример 1   100,00     *1,00
Пример 2   500,00     *0,00
Итого 100,00 руб.
Оплачено 50,00 руб. через терминал CASHMATIC
```

10.3.2.3 Развёрнутый

Пример шаблона при установке программного продукта **NDemia CashMatic KioskBrowser** размещается в файле **"\Program Files\NDemia\CashMatic\PrintTemplates\Развёрнутый.TXT"**
В этом примере показан достаточно сложный шаблон, используются практически все возможности языка описания шаблонов **NDemia CashMatic**. Если Вам нужны начальные сведения о разработке шаблонов, смотрите более простые примеры: "Простой" и "Торговый".

```
%bNDemia CashMatic: %(загол)%b
%d %t
%{если(%(ид)):Код платежа: %(id)
%}%{прод:%(текств)%(номер). %(назв)    %(цена) *%(колво)       = %(стоим)
%i%(текстн)%i
%}%bИтого%b %(итого) руб.
%{cash:%(шрифт2)Оплачено наличными %m руб. через терминал %(терминал)
%{debt:При получении заказа покупатель должен доплатить %(долг) руб.
%}%{change:Покупателю сдача %(сдача) руб.
%}%}%{nocash:Заказ не оплачен
%}%{если(%(перем:ТелефонСлужбыПоддержки)):%{прим:может быть не задан
%}Телефон службы поддержки: %(перем:ТелефонСлужбыПоддержки)
%}
%{фискал:Фискальный режим
%{if(%o):%(штрихкод)%(код)%}%}%{нефискал:%{if(%(код)):Код документа: %o%}%}
```

(•) Пример из комплектации программного продукта может незначительно отличаться от этого текста

При печати:

- %b**NDemia CashMatic: ...**%b - выделение жирным шрифтом строки **"NDemia CashMatic: ..."** (не действует на фискальном регистраторе);
- %(загол) заменяется на CashMatic.Print.Receipt.Title;
- %d заменяется на дату печати чека (в формате краткой даты);
- %t заменяется на время печати чека (в формате времени, без секунд);
- %(ид)/%(id) заменяется на значение CashMatic.Print.Receipt.Id;
- строка **"Код платежа: ..."** (блок %{если**(**%(ид)**):Код платежа:** %(id)**%}**) печатается только в случае, если чек имеет непустой номер или идентификатор (задано свойство CashMatic.Print.Receipt.Id);
- блок **%{**прод**:...%}** при нефискальной печати повторяется для каждой позиции продажи, при создании фискального чека фискальный регистратор сам форматирует строки продажи, текст фискального чека вставляется на место блока **%{**прод**:...%}**;
- внутри блока **%{**прод**:...%}** для каждой позиции продажи (элемента списка продажи - см. объект CashMaticSaleItem):
  - %(текств) - заменяется на CashMaticSaleItem.TextAbove;
  - %(номер) - простой порядковый номер элемента списка, начиная с 1;
  - %(назв) - заменяется на CashMaticSaleItem.Name;
  - %(цена) - заменяется на CashMaticSaleItem.Price (в формате денежной суммы);
  - %(колво) - заменяется на CashMaticSaleItem.Quantity (в формате числа по региональным параметрам операционной системы);
  - %(стоим) - заменяется на вычисляемое значение CashMaticSaleItem.Cost (в формате денежной суммы);
  - %(текстн) - заменяется на CashMaticSaleItem.TextBelow ("%i%(текстн)%i" - то же самое наклонным шрифтом (курсивом));
- %b**Итого**%b - выделение жирным шрифтом слова **"Итого"** (не действует на фискальном регистраторе);
- %(итого) заменяется на CashMatic.Print.Receipt.Total;
- блок **%{**cash**:...%}** (строка **"Оплачено ... руб. через терминал ..."**) печатается только в случае, если были приняты какие-либо наличные;
- %(шрифт2) - выбор шрифта №2 (по таблице прошитых шрифтов используемого оборудования) действует только на фискальном регистраторе (обратите внимание, указание

шрифта работает только в первой позиции выходной строки, хотя в шаблоне эта позиция может быть сдвинута предшествующими синтаксическими конструкциями);

- `%m` заменяется на значение `CashMatic.Print.Receipt.Cash` (в формате денежной суммы);
- `%(терминал)` заменяется на имя терминала, заданное в Панели управления **NDemia CashMatic**;
- блок `%{debt:...%}` (строка **`"При получении заказа покупатель должен доплатить ..."`**) печатается только в случае, если при приёме платежа принято наличных меньше итоговой стоимости чека (см. Чек по шаблону, см. свойство `CashMatic.Print.Receipt.Debt`);
- `%(долг)` заменяется на `CashMatic.Print.Receipt.Debt` (в формате денежной суммы);
- блок `%{change:...%}` (строка **`"Покупателю сдача ..."`**) печатается только в случае, если при приёме платежа принято наличных больше итоговой стоимости чека (см. Чек по шаблону, см. свойство `CashMatic.Print.Receipt.Change`);
- `%(сдача)` заменяется на `CashMatic.Print.Receipt.Change` (в формате денежной суммы);
- блок `%{nocash:...%}` (строка **`"Заказ не оплачен"`**) печатается только в случае, если при приёме платежа наличных принято не было (принципиально возможна печать и нулевых чеков, т.е. без приёма наличных);
- строка **`"Телефон службы поддержки: ..."`** (блок `%{если(%(перем:ТелефонСлужбыПоддержки)): ... %}`) печатается только в случае, если в текущем платёжном сеансе переменная `"ТелефонСлужбыПоддержки"` имеет непустое значение (был сделан вызов `CashMatic.Session.AddProp("ТелефонСлужбыПоддержки", "...")`);
- `%(перем:ТелефонСлужбыПоддержки)` - значение переменной `"ТелефонСлужбыПоддержки"` из текущего платёжного сеанса (как `CashMatic.Session.GetProp("ТелефонСлужбыПоддержки")`);
- блок `%{фискал:...%}` (строка **`"Фискальный режим"`** и последующий опциональный штрих-код) отображается только при печати фискального чека на фискальном регистраторе;
  - `%o`, `%(код)` - заменяется на код документа, назначенный чеку (`CashMatic.Print.Receipt.Code`);
  - `%(штрихкод)` - режим печати штрих-кода, для всей строки, действует только на фискальном регистраторе (обратите внимание, выбор режима штрих-кода работает только в первой позиции выходной строки, хотя в шаблоне эта позиция может быть сдвинута предшествующими синтаксическими конструкциями);
  - строка со штрих-кодом (блок `%{if(%o):%(штрихкод)%(код)%}`) печатается только в случае, если чеку назначен непустой номер документа (задано свойство `CashMatic.Print.Receipt.Code`);
  - код документа на фискальном регистраторе печатается как штрих-код (задано конструкцией "`%(штрихкод)%(код)`");

- блок `%{нефискал:...%}` обрабатывается только при нефискальной печати;
  - строка **`"Код документа: ..."`** (блок `%{if(%(код)):Код документа: %o%}`) печатается только в случае, если чеку назначен непустой номер документа (задано свойство `CashMatic.Print.Receipt.Code`);
  - код документа при нефискальной печати отображается цифрами.

Полный список возможных подставляемых значений: см. Макроимена.
Полный список возможных типов блоков: см. Макроблоки.
Сведения о форматах данных: см. Форматирование данных.
Сведения об управлении шрифтами символов: см. Атрибуты шрифта и Выбор шрифта строки.

Для печати чека по этому шаблону прикладная программа (javascript внутри HTML-страницы) должна задать свойства `Title`, `Id` и `Cash` объекта `CashMatic.Print.Receipt`, в список продажи (объект `CashMatic.Print.Receipt.SaleItems`) нужно добавить хотя бы один элемент (см. объект `CashMaticSaleItem`), для каждого добавленного элемента нужно указать свойства `Name`, `Price` и `Quantity` (опционально можно ещё `TextAbove` и `TextBelow`).
Опционально могут быть заданы свойство `CashMatic.Print.Receipt.Code` и переменная платёжного сеанса `"ТелефонСлужбыПоддержки"` (см. метод `CashMatic.Session.AddProp()`).
Кроме того, нужно указать используемый шаблон и вызвать печать:

```
CashMatic.Print.Receipt.Title="Пример приёма платежа";
CashMatic.Print.Receipt.Id="112";
CashMatic.Print.Receipt.Cash=50;
var Sale1 = CashMatic.Print.Receipt.SaleItems.Add();
Sale1.Name = "Пример 1";
Sale1.Price = 100;
Sale1.Quantity = 1;
var Sale2 = CashMatic.Print.Receipt.SaleItems.Add();
Sale2.Name = "Пример 2";
Sale2.Price = 500;
CashMatic.Print.Receipt.Code="200000000001"; //опционально
CashMatic.Session.AddProp("ТелефонСлужбыПоддержки", "00000000"); //опционально
CashMatic.Print.Receipt.Template="Развёрнутый.txt";
CashMatic.Print.PrintOut();    //при ошибке печати возвращает FALSE
```

Сведения об объекте `CashMatic.Print` и других объектах печати: см. Объекты печати.
Другие примеры подготовки данных для печати чека по шаблону:

- см. Чек по шаблону;
- см. Пример платёжного интерфейса (`finish.html`).

Пример чека, напечатанного по шаблону "Развёрнутый":

```
NDemia CashMatic: пример приёма платежа
04.10.2011 13:20
Код платежа: 00000000
1. Пример 1      100,00      *1,00 = 100,00

2. Пример 2      500,00      *0,00 = 0,00

Итого 100,00 руб.
Оплачено наличными 100,00 руб. через
терминал CASHMATIC
Телефон службы поддержки: 00000000

Код документа: 200000000001
```

11 Звуковая поддержка

NDemia CashMatic KioskBrowser предоставляет разработчику возможность вопроизведения звуковых фонограмм в контексте хост-приложения (в дополнение к имеющимся в **Internet Explorer** средствам озвучивания HTML-документов), звук может воспроизводиться как при переходе со страницы на страницу, так и вообще безотносительно навигации.
Кроме того, предоставляется возможность автоматизированного программного монтажа фонограмм (комбинирование голосовых фраз из звукозаписей отдельных словосочетаний и слов, включая готовое решение для озвучивания чисел, в частности денежных сумм).

Управление списком воспроизведения: методы `Open` и `Add`.
Управление воспроизведением: методы `Play`, `Stop`, `PlayFile`.
Озвучивание чисел и денежных сумм: метод `PlayMoney`.
Воспроизводятся только файлы формата WAV, только с локального диска. (для остальных вариантов прикладной программист может использовать средства **Internet Explorer**, такие, как **`BGSOUND`** и **`t:audio`**).
Если файл находится в папке `Sound` в пути установки **NDemia CashMatic**, то достаточно указывать только его имя, иначе - нужно указывать и путь файла, либо относительно папки `Sound`, либо абсолютный путь.
Если расширение имени файла не задано, то добавляется умолчательное расширение **`".wav"`**.

> **Совместимость:** в версиях **NDemia CashMatic** ранее **2.7.1** в любом случае подставляется расширение **".WAV"**, даже если указано другое.

Несмотря на допустимость любых расширений имён файлов, формат кодирования звуковых файлов всё равно поддерживается только **`WAV`** (waveform audio).

Имя файла может содержать переменные окружения (*environment variables*).

> **Совместимость:** переменные окружения в именах звуковых файлов поддерживаются в **NDemia CashMatic**, начиная с версии **2.7.1**.

Для удобства использования фонограмм рекомендуется имена файлам давать в прямом соответствии с произносимым в них текстом - см. папку **`Sound`**.

11.1 Объект CashMaticSound(CashMatic.Sound), методы

Имя	Тип результата	Тип обращения	Назначение
Add(FileName)	Логический	*Метод*	Добавляет указанный файл в текущий открытый список воспроизведения. Если список уже воспроизводится, то добавленный файл не будет воспроизведён, пока список не будет перезапущен методом **Play**. Возвращает `false` при ошибке.
Open()	Пустой	*Метод*	Открывает новый (пустой) список воспроизведения. Не прерывает текущее воспроизведение, если оно ещё не закончилось.
Play()	Логический	*Метод*	Запускает или перезапускает текущий открытый список воспроизведения. Возвращает `false` при ошибке, при пустом списке или при отключенном звуковом сопровождении.
PlayFile(FileName)	Логический	*Метод*	Сбрасывает текущий список воспроизведения, создаёт новый список из одного файла, прерывает текущее воспроизведение и запускает указанный файл. Возвращает `false` при ошибке или при отключенном звуковом сопровождении.
PlayMoney(Prefix, Money, Suffix)	Логический	*Метод*	Сбрасывает текущий список воспроизведения, создаёт новый список, добавляет в него: 1. файл **Prefix**, если это не пустая строка; 2. комбинацию файлов из папки **Sound**, озвучивающую целое число **Money** как сумму в целых рублях; 3. файл **Suffix**, если это не пустая строка. Прерывает текущее воспроизведение, запускает вновь созданный список. Возвращает `false` при ошибке, или при отключенном звуковом сопровождении, или при отключенном озвучивании денежных сумм.
Stop	Пустой	*Метод*	Прерывает текущее воспроизведение.

12 Дополнительные средства

В состав программного продукта **NDemia CashMatic KioskBrowser** включен ряд дополнительных маленьких программ - команд службы **NDemia CashMatic Kiosk**.
Эти команды предназначены для следующих целей:

- доступ к функциям, реализуемым службой **NDemia CashMatic Kiosk**, из других приложений, в обход объектной модели **NDemia CashMatic** (это возможно и является допустимым поддерживаемым использованием программного продукта **NDemia CashMatic**. Например, известны применения **NDemia CashMatic**, в которых пользовательский интерфейс терминала реализуется локальным бортовым веб-сервером, который сам работает с купюроприёмником и принтером через вызовы команд **NDemia CashMatic Kiosk**);
- отладка прикладных программ, разработанных на основе **NDemia CashMatic** (в том числе скриптов для **NDemia CashMatic KioskBrowser**), на обычных компьютерах - не на платёжных терминалах - симуляция (имитация) приёма купюр, инкассации, отказов и восстановления отказов купюроприёмника.

Если Вы используете приложение **NDemia CashMatic KioskBrowser**, то знание и использование этих команд Вам в целом не нужно. Для отладки прикладных программ могут понадобиться некоторые очень простые вызовы, см. раздел Сокращённые сведения.

Вся функциональность этих команд реализуется службой **NDemia CashMatic Kiosk**, команды лишь обеспечивают интерфейс службы для прикладных программ, поэтому часто они называются интерфейсными командами.

В настоящее время в состав программного продукта **NDemia CashMatic** входят следующие команды:

Команда	Назначение
BVSIMEVENT	симуляция события купюроприёмника
BVSTATMON	отслеживание состояния купюроприёмника
CONFIG	конфигурация различных параметров
CUT	отрезка чека **Совместимость:** команда поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**.
GETCASH	приём купюр
PLAY	вопроизведение фонограмм
PRINTOUT	печать чека (фискальный регистратор не поддерживается)
SIMCASH	симуляция ввода купюры
TICKET	операции с выходным чеком (проверить наличие, втянуть, вытолкнуть) - применима при наличии в принтере соответствующих механизмов удержания/перемещения напечатанного чека. **Совместимость:** команда поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**.

12.1 BVSIMEVENT

см. также:	BVSTATMON	GETCASH	CashMaticCashSession

Команда BVSIMEVENT симулирует событие купюроприёмника (в целях отладки клиентских приложений, например, без физического подключения купюроприёмника). Если имеются ожидающие запросы BVSTATMON с соответствующими масками ожидания, то они завершаются с результатом **6 SIMULATED**.
BVSIMEVENT <event>

<event> - десятичное целое число, код события (см. Коды событий)

Служба хранит в памяти текущую комбинацию флагов состояний в двух экземплярах - реальное состояние и симулированное состояние (см. Флаги состояний). Реальные события влияют на флаги обеих комбинаций одинаково, симулированные события влияют только на флаги симулированного состояния. Если эти комбинации не совпадают, общее состояние считается симулированным, и запрос BVSTATMON с нулевым таймаутом завершается с кодом **6 SIMULATED**.

Симулированные события в лог купюроприёмника не записываются.

Все изменения состояния купюроприёмника, симулируемые командой BVSIMEVENT, обрабатываются соответствующим образом - на них реагируют BVSTATMON, GETCASH, приложение **NDemia CashMatic KioskBrowser** и т.п.

Для выполнения команды BVSIMEVENT в Панели управления **NDemia CashMatic** нужно на вкладке Купюроприёмник установить флажок "Разрешить симуляцию состояния".

Программа BVSIMEVENT.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения в текущей версии никак не индицируется и не возвращается.
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова BVSIMEVENT.

12.2 BVSTATMON

см. также:	BVSIMEVENT	GETCASH	CashMaticCashSession

Команда BVSTATMON обеспечивает клиентской программе возможность отслеживать различные состояния и переходы состояний купюроприёмника.
Основной целью является получение сигнала о неисправности или вскрытии купюроприёмника, а также получение сигнала о восстановлении работоспособности купюроприёмника.

BVSTATMON [<respfile> [<mask> [<timeout>]]]

<respfile> - имя файла ответа, в ResponseDir;
<mask> - битовая маска ожидаемых состояний (см. Флаги состояний), шестнадцатеричное число (только HEX, без дополнительных спецификаций)
<timeout> - таймаут ожидания ввода, секунды (десятичное число)
Ответ записывается в файл, первая строка может иметь одно из перечисленных значений:
0 OK
1 TIMEOUT
4 CANCEL ;запрос прерван (программно - либо BVSTATMON без параметров, см ниже, либо завершение работы службы или системы)
6 SIMULATED ;состояние симулировано, см. команду BVSIMEVENT. По сути этот результат полностью аналогичен **0 OK**, однако возвращается другим кодом, чтобы предоставить клиентской программе возможность отличать смоделированные события от реальных.
Если первая строка **0 OK** или **6 SIMULATED**, то вторая - битовая комбинация полученных состояний (шестнадцатеричное число - только HEX)

При нулевом/отсутствующем таймауте ожидание не выполняется, результатом может быть либо **0 OK**/**6 SIMULATED**, либо **1 TIMEOUT**.
Вызов с нулевым таймаутом используется для прямого чтения текущего состояния.

Команда BVSTATMON без параметров прерывает (**4 CANCEL**) сразу все ожидающие конкурирующие запросы.

Вызывающее приложение может узнать о готовности ответа по завершению процесса BVSTATMON.EXE

Маска ожидаемых состояний может иметь любое число бит (0..32).

> ***Примечание:*** *приложение* ***NDemia CashMatic KioskBrowser*** *имеет собственный встроенный код, полностью соответствующий интерфейсу BVSTATMON, поэтому описание данной команды является действительной документацией по механизму мониторинга купюроприёмника для всего продукта* ***NDemia CashMatic****.*

Программа BVSTATMON.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения записывается в текстовый файл под указанным именем в пути ResponseDir (конфигурируемый параметр службы **NDemia CashMatic Kiosk**, настраивается командой CONFIG).
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова BVSTATMON.

Определён фиксированный набор событий, которые может распознать служба **NDemia CashMatic Kiosk**. Каждому событию сопоставлены код события и флаг состояния.

Когда некоторое событие происходит, служба делает запись с соответствующим кодом события в логе купюроприёмника. При этом устанавливается соответствующий флаг в комбинации состояний, хранящейся в памяти службы, кроме того, в этой комбинации могут быть сброшены некоторые другие флаги состояний, в зависимости от конкретного события. Таким образом, установленное значение флага означает, что когда-то произошло событие, устанавливающее это состояние, и после того события не произошло событие, сбрасывающее это состояние.

Например, установление связи с купюроприёмником фиксируется в логе купюроприёмника как событие ONLINE, при этом служба устанавливает флаг состояния ONLINE. После этого связь может сохраняться много суток, в лог может быть записано много других событий, однако флаг будет оставаться установленным до тех пор, пока не произойдёт событие OFFLINE, поэтому клиентская программа с помощью BVSTATMON в любой момент может определить текущее состояние связи.

BVSTATMON возвращает позитивный результат (**0 OK**/**6 SIMULATED**) если/когда установлен хотя бы один флаг из маски ожидания.

С помощью команды BVSIMEVENT флаги состояний можно устанавливать принудительно. При этом:

записи в лог купюроприёмника не делаются;
сброс флагов соответствующим образом связанных состояний выполняется (в том числе принудительно установленных);
соответствующие по маске ожидания активные запросы BVSTATMON срабатывают.

Служба отслеживает, какие флаги состояний были установлены принудительно, поэтому запросы BVSTATMON всегда могут получить соответствующий действительности результат **0 OK** или **6 SIMULATED**.

Запросов BVSTATMON может выполняться одновременно несколько (в отличие от GETCASH), например, с разными масками. Это следует учесть в части задания имён файлов - одно имя не должно использоваться в нескольких запросах, иначе возможно наложение ответов (служба формирует файлы ответов в режиме **shared read-write-delete**, поэтому наложение ответов действительно возможно и не вызывает конфликта с точки зрения файлового доступа).

Служба открывает файлы ответов в режиме **FILE_SHARE_DELETE**, что позволяет вызывающей программе перед вызовом службы создать файл с флагом **FILE_FLAG_DELETE_ON_CLOSE** (см. документацию по **Windows API**, функция **CreateFile**), и в дальнейшем не заботиться об удалении этого файла - операционная система автоматически удалит его даже при сбоях работы программного обеспечения.

12.2.1 Лог купюроприёмника

Лог купюроприёмника:
настройка пути - см. CONFIG BillValidatorLogDir (указывается папка, в ней создаются файлы на каждые сутки, имена в формате ГГГГ-ММ-ДД.txt)
все строки имеют одинаковый формат
<дата-время><таб><код события><таб><мнемоника><таб>;<комментарий>
симулированные события (см. команду BVSIMEVENT) в лог не записываются.

Код события (десятичное число)	Мнемоника	Примечание
1	LOGOPEN	лог открыт
2	LOGCLOSE	лог закрыт
3	START	запуск сервиса
4	STOP	остановка сервиса
5	CONFIG	конфигурация сервиса
6	OFFLINE	нет связи с купюроприёмником (возможно, таймаут)
7	ONLINE	купюроприёмник на связи
8	RESET	сброс купюроприёмника
9	READY	купюроприёмник готов к работе (красная лампа)
10	ENABLED	ввод купюр разрешён (зелёная лампа)
11	REJECT	отказ (возврат купюры)
12	STACK	купюра принята
13	FULL	переполнение стекера
14	DISPOSED	стекер удалён
15	CHEATED	попытка вытащить купюру из купюроприёмника во время её загрузки или проверки.
16	FAILURE	отказ оборудования, замятие купюры и т.п.
17	UNKNOWN	неизвестный (недокументированный) статус купюроприёмника
18	RESTORE	восстановление после ошибки
19	DETECT	купюра опознана и возвращена, положительный результат в режиме проверки купюр **Совместимость:** код добавлен в **NDemia CashMatic** версии **2.6.0**
20	PAUSE	купюры запутались, предположительно слишком частая вставка купюр друг за другом *(только в CCNET [•])*. **Совместимость:** код добавлен в **NDemia CashMatic** версии **2.6.0**
21	RETURN	возврат купюры: купюра опознана, но номинал купюры выходит за пределы ограничения диапазона допустимых номиналов купюр, см. объект **CashMaticCashSession**, свойства **MinBill**, **MaxBill**, метод **SetBillRange**. **Совместимость:** код добавлен в **NDemia CashMatic** версии **2.7.0**

(•) событие **PAUSE** возможно только при использовании протокола **CCNET**. В соответствии с рекомендациями протокола CCNET при этом событии запрашивается сброс устройства.

12.2.2 Флаги состояний

Флаги состояний однозначно соответствуют кодам событий. Коды событий равны кодам записей лога купюроприёмника.
Каждое событие добавляет в комбинацию флагов состояний свой флаг и сбрасывает некоторые другие флаги.

Код события (десятичное число)	Мнемоника события	Маска события (16-ричное число)	Событие сбрасывает флаги
1	LOGOPEN	00000001	LOGCLOSE
2	LOGCLOSE	00000002	LOGOPEN
3	START	00000004	STOP, CONFIG, RESET, READY, ENABLED, REJECT, STACK, FULL, DISPOSED, CHEATED, FAILURE, UNKNOWN, RESTORE, DETECT, PAUSE, RETURN
4	STOP	00000008	START, CONFIG, RESET, READY, ENABLED, REJECT, STACK, FULL, DISPOSED, CHEATED, FAILURE, UNKNOWN, RESTORE, DETECT, PAUSE, RETURN
5	CONFIG	00000010	-
6	OFFLINE	00000020	ONLINE, RESET, READY, ENABLED, REJECT, STACK, FULL, DISPOSED, CHEATED, FAILURE, UNKNOWN, RESTORE, DETECT, PAUSE, RETURN
7	ONLINE	00000040	OFFLINE, RESTORE, PAUSE
8	RESET	00000080	OFFLINE *(только в CCNET (•))*, READY, ENABLED, REJECT, STACK, CHEATED, FAILURE, RESTORE, DETECT, RETURN
9	READY	00000100	OFFLINE, RESET, ENABLED, REJECT, STACK, DISPOSED, CHEATED, FAILURE *(только в CCNET (••))*, FULL, DETECT, PAUSE, RETURN
10	ENABLED	00000200	OFFLINE, RESET, REJECT, STACK, DISPOSED, DETECT, PAUSE, RETURN
11	REJECT	00000400	OFFLINE, RESET, STACK, RESTORE, DETECT, PAUSE, RETURN
12	STACK	00000800	OFFLINE, RESET, REJECT, FULL, DISPOSED, RESTORE, DETECT, PAUSE, RETURN
13	FULL	00001000	OFFLINE, RESET, DISPOSED, RESTORE, READY, PAUSE
14	DISPOSED	00002000	OFFLINE, RESET, READY, ENABLED, STACK, FULL, RESTORE, DETECT, PAUSE, RETURN
15	CHEATED	00004000	PAUSE

16	FAILURE	00008000	READY, RESTORE, PAUSE
17	UNKNOWN	00010000	PAUSE
18	RESTORE	00020000	FAILURE
19	DETECT	00040000	OFFLINE, RESET, REJECT, STACK, RESTORE, PAUSE, RETURN
20	PAUSE	00080000	-
21	RETURN	00100000	-

(•) флаг **OFFLINE** сбрасывается событием **RESET** только при использовании протокола **CCNET**, поскольку сброс купюроприёмника по этому протоколу выполняется с подтверждением со стороны устройства, таким образом, успешное выполнение сброса однозначно указывает на наличие подключенного работоспособного устройства.
(••) флаг **FAILURE** сбрасывается событием **READY** только при использовании протокола **CCNET**, поскольку по этому протоколу готовность купюроприёмника означает отсутствие проблем, и не предусмотрено отдельных оповещений об устранении предшествовавших отказов.

Совместимость: коды **DETECT**, **PAUSE** поддерживаются в **NDemia CashMatic**, начиная с версии **2.6.0**

Совместимость: код **RETURN** поддерживается в **NDemia CashMatic**, начиная с версии **2.7.0**

12.3 CONFIG

Команда CONFIG конфигурирует различные параметры службы **NDemia CashMatic Kiosk** без необходимости перезапуска службы.
Следует заметить, что практически все параметры, которые вообще имеет смысл менять, могут управляться через Панель управления **NDemia CashMatic**. Команда CONFIG предназначена главным образом для пакетного применения и обращений к службе из других программ.
Программа CONFIG.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения в текущей версии никак не индицируется и не возвращается.
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова CONFIG.

Основной синтаксис командной строки при вызове:
CONFIG <ИмяПараметра>=<Значение>

Верхний/нижний регистр букв в имени параметра не различается.
Успешно переданные значения сразу же заносятся в реестр, поэтому запускать CONFIG нужно только для изменения настройки.

CONFIG BillValidatorBadStatusMask=<mask>

<mask> - комбинация флагов нештатного ("плохого") состояния купюроприёмника, 16-ричное число (см. Флаги состояния).
Данное значение используется для мониторинга состояния оборудования (купюроприёмника) при вводе купюр (см. описание команды GETCASH, см. также описание объекта CashMaticCashSession). Если на момент запуска ожидания ввода купюры установлен какой-либо из флагов комбинации **BillValidatorBadStatusMask**, то ожидание немедленно завершается:

- команда GETCASH завершается с результатом **5 INTERRUPT**;
- активный сеанс купюроприёмника (см. объект CashMaticCashSession) завершается событием OnCancel.

Маской **BillValidatorBadStatusMask** проверяются только флаги, уже установленные на момент запуска ожидания ввода купюры, на события, происходящие во время ожидания, эта маска не распространяется (может быть изменено в будущих версиях).
По умолчанию (при установке **NDemia CashMatic**) устанавливается значение **0xB008 (FAILURE | DISPOSED | FULL | STOP)**. Рекомендуется не изменять это значение без существенных причин.
Если в момент вызова **CONFIG BillValidatorBadStatusMask** уже происходит ожидание ввода купюры (выполняется GETCASH или активирован объект CashMaticCashSession), и текущее состояние купюроприёмника по новой маске становится нештатным ("плохим"), то текущее ожидание купюры прерывается, аналогично нештатному состоянию в момент начала ожидания:

- команда GETCASH завершается с результатом **5 INTERRUPT**;

> **Совместимость:** при использовании протокола купюроприёмника **ICT004** в версиях **NDemia CashMatic** ранее **2.7.1** результат в этом случае: **3 CANCEL**

- активный сеанс купюроприёмника (см. объект CashMaticCashSession) завершается событием OnCancel.

CONFIG BillValidatorLogDir=<path>

<path> - полный путь директории, куда записывать лог купюроприёмника (путь требуется указывать именно полный, поскольку текущие диск/директория в этом случае не определены)
Имена файлов лога купюроприёмника формируются как **ГГГГ-ММ-ДД.txt**

CONFIG BillValidatorPort=<port>

<port> - порт купюроприёмника, например **COM1**

CONFIG BillSecurityMask=<mask>

<mask> - маска повышенных требований ("BILL SECURITY MASK", только CCNET), 16-ричная, единичные биты соответствуют типам купюр, к которым должны применяться повышенные требования.

Обычное соответствие типов купюр номиналам для российских рублей:

Тип купюры	Номинал, рублей	Бит маски повышенных требований (16-ричное число)
1 (•)	5	02
2	10	04
3	50	08
4	100	10
5	500	20
6	1000	40
7	5000	80

(•) обычно 5 рублей купюроприёмники не принимают, тем не менее код для такого типа купюр зарезервирован.

CONFIG PrinterBaud=<value>

<value> - скорость порта принтера, бит/с.

CONFIG PrinterCommands=<filepath>

<filepath> - полный путь файла описания команд принтера (подробности см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt) (путь требуется указывать именно полный, поскольку текущие диск/директория в этом случае не определены)

CONFIG PrinterConvertOem=<value>

<value> - если ненулевое десятичное число, то включается функция преобразования кодировки текста при печати "CP1251(Windows)->CP866(DOS)" - требуется для некоторых моделей принтеров.

По умолчанию преобразование кодировки выключено.

Преобразование выполняется как при печати, так и в режиме вывода в файл (см. CONFIG PrinterFile).

CONFIG PrinterFile=<filepath>

<filepath> - полный путь файла вывода принтера (путь требуется указывать именно полный, поскольку текущие диск/директория в этом случае не определены)

Вывод печати в файл может использоваться, например, в целях отладки клиентских приложений (без физического подключения принтера)

Если задан файл вывода принтера, то чек не печатается, а дописывается в конец этого файла. Статус принтера при этом не проверяется (исключение: см. CONFIG PrinterSimulateNoPaper), все коды управления и форматирования записываются в соответствии текущими настройками (см. CONFIG PrinterFormat и CONFIG PrinterCommands)

Если в качестве **<filepath>** задана пустая строка, то задаётся режим "принтер не подключен" (любой вызов PRINTOUT завершается с результатом **2 OFFLINE**).

CONFIG PrinterFormat=<value>

<value> - если ненулевое десятичное число, то включается форматирование шрифта при печати (обработка %-кодов, см. команду PRINTOUT)
По умолчанию форматирование включено.
Если форматирование выключено, то все %-коды в шаблоне чека пропускаются без обработки (не удаляются)

CONFIG PrinterPort=<port>

<port> - порт принтера, например COM2
Если в качестве **<port>** задана пустая строка, то задаётся режим "принтер не подключен" (любой вызов PRINTOUT завершается с результатом **2 OFFLINE**).

CONFIG PrinterSimulateNoPaper=<value>

<value> - если ненулевое десятичное число, то включается симуляция отсутствия бумаги в принтере (для отладки клиентских приложений).
По умолчанию симуляция отсутствия бумаги отключена.
Симуляция отсутствия бумаги может выполняться в режиме вывода в файл (см. CONFIG PrinterFile)
Симуляция отсутствия бумаги не выполняется в следующих случаях:
если отключена проверка статуса принтера (см. CONFIG PrinterStatusCheck);
если задан режим "принтер не подключен" (см. CONFIG PrinterPort, CONFIG PrinterFile).

CONFIG PrinterStatusCheck=<value>

<value> - если ненулевое десятичное число, то включается проверка статуса принтера перед печатью.
По умолчанию проверка статуса принтера включена.
В режиме вывода в файл (см. CONFIG PrinterFile) статус принтера не проверяется при любом значении PrinterStatusCheck (за исключением симуляции отсутствия бумаги - см. CONFIG PrinterSimulateNoPaper).

CONFIG ResponseDir=<path>

<path> - полный путь директории, куда записывать файлы ответов (путь требуется указывать именно полный, поскольку текущие диск/директория в этом случае не определены).
Имена файлов ответов задаются в параметрах запросов (см. команды BVSTATMON, CUT, GETCASH, PRINTOUT, TICKET).
По умолчанию используется путь
"\Documents and Settings\All Users\Application Data\NDemia\CashMatic\ResponseDir" (**Windows 2000**, **Windows XP**);
или
"\ProgramData\NDemia\CashMatic\ResponseDir" (**Windows Vista**, **Windows 7**).

12.4 CUT

Команда CUT выполняет отрезку напечатанного чека.

Совместимость: команда поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**.

CUT <respfile>

<respfile> - имя файла ответа, в ResponseDir;
Ответ записывается в файл, первая строка может иметь одно из перечисленных значений:
0 OK ;Чек отрезан (на второй строке двузначное 16-ричное число, комбинация флагов состояния рулона, полученных по команде STATUS_ROLL (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt): 1 - сработал датчик приближения конца бумаги, 2 - сработал датчик отсутствия рулона, ноль означает нормальное состояние или неизвестное состояние.
2 OFFLINE ;нет связи с принтером, или принтер не работает (диагностика на второй строке, двузначное 16-ричное число, равно статусу принтера, полученному по команде STATUS_PRINT (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt), ноль означает невозможность получения статуса, например, из-за ошибки связи)
Нулевой код диагностики **OFFLINE** означает отсутствие принтера (по крайней мере, ошибку связи с принтером или ошибку настройки подключения).
Ненулевой код диагностики **OFFLINE** соответствует возвращённому принтером коду состояния, интерпретация значения зависит от модели принтера.
В любом случае, с точки зрения интерфейса приёма платежей, **2 OFFLINE** означает неисправность принтера, не связанную с отсутствием бумаги.
3 NO PAPER
В принтере кончилась бумага.
Для отладки клиентских приложений можно "спровоцировать" результат **3 NO PAPER** - см. CONFIG PrinterSimulateNoPaper (если задать ненулевое значение PrinterSimulateNoPaper, то любой вызов PRINTOUT будет завершаться с результатом **3 NO PAPER**)
4 CANCEL - не настроена система команда принтера, либо файловый режим принтера (см. вкладку Принтер в Панели управления **NDemia CashMatic**). Вторая строка ответа формируется так же, как для ответа **0 OK**.
пустой ответ (команда завершилась, но не вернула файл ответа или вернула пустой файл ответа) - команда не поддерживается в данной версии службы **NDemia CashMatic Kiosk**.

Программа CUT.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения записывается в текстовый файл под указанным именем в пути ResponseDir (конфигурируемый параметр службы **NDemia CashMatic Kiosk**, настраивается командой CONFIG).
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова CUT.

Служба открывает файлы ответов и файлы данных в режиме **FILE_SHARE_DELETE**, что позволяет вызывающей программе перед вызовом службы создать файл с флагом **FILE_FLAG_DELETE_ON_CLOSE** (см. документацию по **Windows API**, функция **CreateFile**), и в дальнейшем не заботиться об удалении этого файла - операционная система автоматически удалит его даже при сбоях работы программного обеспечения.

Вызывающее приложение может узнать о готовности ответа по завершению процесса CUT.EXE.

12.5 GETCASH

см. также:	BVSTATMON	SIMCASH	CashMaticCashSession

Команда GETCASH вызывает (или прерывает) полный цикл действий по приёму купюры (включая контроль таймаута ожидания).

GETCASH [<respfile> [<timeout>]]

<respfile> - имя файла ответа, в ResponseDir;

<timeout> - таймаут ожидания ввода, секунды (по умолчанию ноль).

Ответ записывается в файл, первая строка может иметь одно из перечисленных значений:

0 OK ;Купюра принята, код купюры - на второй строке

1 TIMEOUT ;Таймаут истёк, купюра не принята

2 OFFLINE ;Нет связи с купюроприёмником

3 BUSY ;Купюроприёмник занят конкурирующим запросом GETCASH (многозадачное обращение допустимо)

4 CANCEL ;Запрос прерван (программно - либо GETCASH без параметров, см ниже, либо завершение работы службы или системы)

5 INTERRUPT ;Запрос прерван по состоянию купюроприёмника (например, вынули стекер)

6 SIMULATED ;Ввод купюры симулирован, код купюры - на второй строке (см. команду SIMCASH). По сути этот результат полностью аналогичен **0 OK**, однако возвращается другим кодом, чтобы предоставить клиентской программе возможность отличать смоделированные события от реальных.

Если первая строка **0 OK** или **6 SIMULATED**, то вторая - тип принятой купюры (число).
Если тип принятой купюры равен нулю, то это означает, что произошёл возврат купюры пользователю. Возврат делается в случае браковки купюры при приёме/проверке в сеансе купюроприёмника (см. объект CashMaticCashSession). Номинал купюры недоступен (либо купюроприёмник его не определил, либо определил как недопустимый).

Если тип купюры не равен нулю, то его можно преобразовать в номинал купюры по таблице соответствия купюр.
Обычное соответствие типов купюр номиналам для российских рублей:

Тип купюры	Номинал, рублей
1 (•)	5
2	10
3	50
4	100
5	500
6	1000
7	5000

(•) обычно 5 рублей купюроприёмники не принимают, тем не менее код для такого типа купюр зарезервирован.

При нулевом таймауте ожидание ввода купюр не выполняется. В этом случае, при отсутствии проблем оборудования и связи, результатом может быть либо **"1 TIMEOUT"**, либо **"3 BUSY"**. Результат **"3 BUSY"** указывает на наличие конкурирующего запроса (вызов с нулевым таймаутом может быть использован для обнаружения конкурирующего запроса, сам по себе такой вызов гарантированно не создаёт конкуренции другим запросам).

Имя файла ответа рекомендуется формировать при каждом запросе уникальное, поскольку запросов может быть вызвано одновременно несколько (все, кроме первого, получат результат **3 BUSY**, но тем не менее эта ситуация возможна). Служба формирует файлы ответов в режиме **shared read-write-delete**, поэтому наложение ответов действительно возможно и не вызывает конфликта с точки зрения файлового доступа.

Если команда вызвана без параметров, то она прерывает ожидание конкурирующего запроса с результатом **4 CANCEL**. Клиентское приложение может использовать это для досрочного принудительного завершения предыдущего вызова GETCASH.

> ***Примечание:*** *при использовании* ***NDemia CashMatic KioskBrowser*** *вызов GETCASH без параметров приводит к тому, что текущий активный сеанс купюроприёмника (см. объект CashMaticCashSession) завершается событием OnCancel.*

Команда GETCASH может реагировать на отказы, которые произошли до её вызова. Состояние купюроприёмника определяется по установленным на момент вызова флагам (см. Флаги состояний), учитываются флаги, задаваемые командой CONFIG BillValidatorBadStatusMask, если какие-либо из флагов установлены (в том числе с помощью команды BVSIMEVENT), то GETCASH немедленно завершается с результатом **5 INTERRUPT**.

Программа GETCASH.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения записывается в текстовый файл под указанным именем в пути ResponseDir (конфигурируемый параметр службы **NDemia CashMatic Kiosk**, настраивается командой CONFIG).
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова GETCASH.

Служба открывает файлы ответов в режиме **FILE_SHARE_DELETE**, что позволяет вызывающей программе перед вызовом службы создать файл с флагом **FILE_FLAG_DELETE_ON_CLOSE** (см. документацию по **Windows API**, функция **CreateFile**), и в дальнейшем не заботиться об удалении этого файла - операционная система автоматически удалит его даже при сбоях работы программного обеспечения.

Вызывающее приложение может узнать о готовности ответа по завершению процесса GETCASH.EXE

12.6 PLAY

Команда PLAY воспроизводит звукозаписи (только WAV-формат).

```
PLAY [<file1>[ ... <fileN>]]
```

<file1>..<fileN> - имена (список имён) файлов, содержащих воспроизводимые фонограммы. Основным местом хранения фонограмм (звуковых файлов) является папка **Sound** в пути установки продукта (обычно **"\Program Files\NDemia\CashMatic\Sound"**). При вызове команды PLAY можно указывать имена файлов без пути, с путём относительно папки **Sound**, или с полным абсолютным путём.

Совместимость: в **NDemia CashMatic** версий ранее **2.7.1** для каждого файла обязательно требуется указывать полный абсолютный путь, с явным указанием расширения **.wav**.

Выполнение команды PLAY завершается без ожидания окончания воспроизведения звука, команда только помещает фонограммы в очередь воспроизведения.
Если предыдущее воспроизведение не завершено, новый вызов PLAY прерывает (принудительно завершает) предыдущее воспроизведение.
Вызов PLAY без параметров просто прерывает текущее вопроизведение.
Вызов команды PLAY с несколькими файлами существенно отличается от вызова PLAY несколько раз с разными файлами, поскольку при воспроизведении списка файлов служба **NDemia CashMatic Kiosk** обеспечивает воспроизведение каждого файла до конца (возможность прерывания другим вызовом остаётся).

Пример вызова трёх файлов (формируется фраза "Одна тысяча рублей"):

```
PLAY одна тысяча рублей
```

Такой вызов обеспечивает последовательное воспроизведение фонограмм **"\Program Files\NDemia\CashMatic\Sound\Одна.wav"**, **"\Program Files\NDemia\CashMatic\Sound\Тысяча.wav"**и **"\Program Files\NDemia\CashMatic\Sound\Рублей.wav"**.

Другой пример (имена файлов содержат пробелы):

```
PLAY "Терминал готов к работе" "Введите код платежа"
```

> ***Примечание:*** *флажок* ***"Использовать звуковые сообщения"*** *на вкладке Интерфейс в Панели управления* ***NDemia CashMatic*** *не влияет на возможность вопроизведения фонограмм командой PLAY (снятие флажка не отключает звук в PLAY).*

Программа PLAY.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения в текущей версии никак не индицируется и не возвращается.
При вызове PLAY служба **NDemia CashMatic Kiosk** автоматически запускается (если не работала до того).

Совместимость: в **NDemia CashMatic** версий ранее **2.7.2** служба **NDemia CashMatic Kiosk** должна работать на момент вызова PLAY.

12.7 PRINTOUT

Команда PRINTOUT выполняет полный цикл действий по печати на чековый принтер.

PRINTOUT <respfile> <data>

<respfile> - имя файла ответа, в ResponseDir;
<data> - имя выводимого файла, требуется указывать полный (абсолютный) путь, может отсутствовать - см. ниже.
Ответ записывается в файл, первая строка может иметь одно из перечисленных значений:
0 OK ;Чек напечатан (на второй строке двузначное 16-ричное число, комбинация флагов состояния рулона, полученных по команде STATUS_ROLL (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt): 1 - сработал датчик приближения конца бумаги, 2 - сработал датчик отсутствия рулона, ноль означает нормальное состояние или неизвестное состояние.
1 BAD FILE ;не удалось открыть входной файл, или ошибка чтения
2 OFFLINE ;нет связи с принтером, или принтер не работает (диагностика на второй строке, двузначное 16-ричное число, равно статусу принтера, полученному по команде STATUS_PRINT (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt), ноль означает невозможность получения статуса, например, из-за ошибки связи)
Нулевой код диагностики **OFFLINE** означает отсутствие принтера (по крайней мере, ошибку связи с принтером или ошибку настройки подключения).
Ненулевой код диагностики **OFFLINE** соответствует возвращённому принтером коду состояния, интерпретация значения зависит от модели принтера.
В любом случае, с точки зрения интерфейса приёма платежей, **2 OFFLINE** означает неисправность принтера, не связанную с отсутствием бумаги.
3 NO PAPER
В принтере кончилась бумага.
Для отладки клиентских приложений можно "спровоцировать" результат **3 NO PAPER** - см. CONFIG PrinterSimulateNoPaper (если задать ненулевое значение PrinterSimulateNoPaper, то любой вызов PRINTOUT будет завершаться с результатом **3 NO PAPER**)

Если файл данных для печати не задан, то печать не выполняется, но проверка состояния принтера выполняется, т.е. возможен результат **0 OK**, **2 OFFLINE**, **3 NO PAPER**.

При печати выполняется управление атрибутами шрифта.

Программа PRINTOUT.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения записывается в текстовый файл под указанным именем в пути ResponseDir (конфигурируемый параметр службы **NDemia CashMatic Kiosk**, настраивается командой CONFIG).
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова PRINTOUT.

Служба открывает файлы ответов и файлы данных в режиме **FILE_SHARE_DELETE**, что позволяет вызывающей программе перед вызовом службы создать файл с флагом **FILE_FLAG_DELETE_ON_CLOSE** (см. документацию по **Windows API**, функция **CreateFile**), и в дальнейшем не заботиться об удалении этого файла - операционная система автоматически удалит его даже при сбоях работы программного обеспечения.

Вызывающее приложение может узнать о готовности ответа по завершению процесса PRINTOUT.EXE

12.8 SIMCASH

см. также:	BVSIMEVENT	GETCASH	CashMaticCashSession

Команда SIMCASH симулирует ввод купюры в купюроприёмник (в целях отладки клиентских приложений, например, без физического подключения купюроприёмника, см. также BVSIMEVENT)

SIMCASH <type>

<type> - десятичное целое число, тип купюры. Нулевой тип (**SIMCASH 0**) воспринимается как возврат купюры.

Если имеется ожидающий запрос GETCASH, то он завершается с результатом **6 SIMULATED**.

> ***Примечание:*** *если приём наличных денег выполняется приложением* ***NDemia CashMatic KioskBrowser****, то в результате вызова SIMCASH текущий активный сеанс купюроприёмника (см. объект CashMaticCashSession) завершается событием OnAccept, OnDetect или OnReject.*

Программа SIMCASH.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения в текущей версии никак не индицируется и не возвращается.
Служба **NDemia CashMatic Kiosk** должна работать на момент вызова SIMCASH.

Для выполнения команды SIMCASH в Панели управления **NDemia CashMatic** нужно на вкладке Купюроприёмник установить флажок "Разрешить симуляцию ввода купюр". Кроме того, если используется приложение **NDemia CashMatic KioskBrowser**, то на вкладке Интерфейс нужно снять флажок "Игнорировать симуляцию ввода купюр".

12.9 TICKET

Команда TICKET выполняет операции с выходным чеком (проверить наличие, втянуть, вытолкнуть), команда применима при наличии в принтере соответствующих механизмов удержания/перемещения напечатанного чека (т.н. презентер, ретрактор, ejector и т.п.).

Совместимость: команда поддерживается в **NDemia CashMatic**, начиная с версии **2.6.0**.

TICKET <respfile> [subcommand]

<respfile> - имя файла ответа, в ResponseDir;

Ответ записывается в файл, первая строка может иметь одно из перечисленных значений:

0 OK ;операция выполнена с положительным результатом (на второй строке двузначное 16-ричное число, комбинация флагов состояния рулона, полученных по команде STATUS_ROLL (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt): 1 - сработал датчик приближения конца бумаги, 2 - сработал датчик отсутствия рулона, ноль означает нормальное состояние или неизвестное состояние.

2 OFFLINE ;нет связи с принтером, или принтер не работает (диагностика на второй строке, двузначное 16-ричное число, равно статусу принтера, полученному по команде STATUS_PRINT (см. комментарии входящих в комплект программного продукта файлов PrinterCommands*.txt), ноль означает невозможность получения статуса, например, из-за ошибки связи)

Нулевой код диагностики **OFFLINE** означает отсутствие принтера (по крайней мере, ошибку связи с принтером или ошибку настройки подключения).

Ненулевой код диагностики **OFFLINE** соответствует возвращённому принтером коду состояния, интерпретация значения зависит от модели принтера.

В любом случае, с точки зрения интерфейса приёма платежей, **2 OFFLINE** означает неисправность принтера, не связанную с отсутствием бумаги.

3 NO PAPER

В принтере кончилась бумага.

Для отладки клиентских приложений можно "спровоцировать" результат **3 NO PAPER** - см. CONFIG PrinterSimulateNoPaper (если задать ненулевое значение PrinterSimulateNoPaper, то любой вызов PRINTOUT будет завершаться с результатом **3 NO PAPER**)

4 CANCEL - отменено - либо нет чека на выходе, либо при определении наличия чека отключен опрос принтера, либо не настроена система команда принтера, либо файловый режим принтера (см. вкладку Принтер в Панели управления **NDemia CashMatic**). Вторая строка ответа формируется так же, как для ответа **0 OK**.

пустой ответ (команда завершилась, но не вернула файл ответа или вернула пустой файл ответа) - команда не поддерживается в данной версии службы **NDemia CashMatic Kiosk**.

<subcommand> - **PRESENT** (по умолчанию), **EJECT** или **RETRACT**.

PRESENT - проверка наличия выходного чека;

EJECT - вытолкнуть чек;

RETRACT - втянуть чек.

возвращаемый результат (**<respfile>**) во всех случаях имеет одинаковый синтаксис.

если все команды настроены, но отключен опрос принтера, то TICKET <respfile> PRESENT вернёт отрицательный результат, а TICKET <respfile> EJECT и TICKET <respfile> RETRACT выполнятся с положительным результатом ("**0 OK**").

если все команды настроены и опрос принтера работает, а чек на выходе презентера отсутствует, то TICKET <respfile> EJECT и TICKET <respfile> RETRACT будут завершаться отрицательно ("**4 CANCEL**"), без подачи соответствующих команд принтеру.

Программа TICKET.EXE представляет собой безоконный (не консольный и не скрыто-оконный) процесс операционной системы, результат выполнения записывается в текстовый файл под указанным именем в пути ResponseDir (конфигурируемый параметр службы **NDemia CashMatic Kiosk**, настраивается командой CONFIG).

Служба **NDemia CashMatic Kiosk** должна работать на момент вызова TICKET.

Служба открывает файлы ответов и файлы данных в режиме **FILE_SHARE_DELETE**, что позволяет вызывающей программе перед вызовом службы создать файл с флагом **FILE_FLAG_DELETE_ON_CLOSE** (см. документацию по **Windows API**, функция **CreateFile**), и в дальнейшем не заботиться об удалении этого файла - операционная система автоматически удалит его даже при сбоях работы программного обеспечения.

Вызывающее приложение может узнать о готовности ответа по завершению процесса TICKET.EXE.

12.10 Сокращённые сведения

В этом разделе перечислены команды, которые могут понадобиться для отладки прикладных программ, разработанных на основе **NDemia CashMatic KioskBrowser**.
Команды вводятся с командной строки (можно также создать командные сценарии или вызывать их из других программ). EXE-файлы располагаются в пути установки программного продукта (**"\Program Files\NDemia\CashMatic"**)
Вся функциональность этих команд реализуется службой **NDemia CashMatic Kiosk**, команды лишь обеспечивают интерфейс службы, поэтому для их выполнения служба должна быть запущена (см. Панель управления Windows/Администрирование/Службы)

`SIMCASH <type>`

симуляция (имитация) ввода купюры
<type> - тип купюры (десятичное число)
см. описание команды SIMCASH.
Обычное соответствие типов купюр номиналам для российских рублей:

Вызов	Значение
SIMCASH 0	возврат купюры
SIMCASH 1	5 р. (обычно 5 рублей купюроприёмники не принимают, тем не менее код для такого типа купюр зарезервирован)
SIMCASH 2	10 р.
SIMCASH 3	50 р.
SIMCASH 4	100 р.
SIMCASH 5	500 р.
SIMCASH 6	1000 р.
SIMCASH 7	5000 р.

Для выполнения команды SIMCASH в Панели управления **NDemia CashMatic** нужно на вкладке Купюроприёмник установить флажок "Разрешить симуляцию ввода купюр" и на вкладке Интерфейс снять флажок "Игнорировать симуляцию ввода купюр".

`BVSIMEVENT <event>`

симуляция (имитация) события сервиса купюроприёмника.
<event> код события (десятичное число)
см. описание команды BVSIMEVENT
Коды событий, симуляция которых может потребоваться для отладки прикладной программы

Вызов	Значение
BVSIMEVENT 9	**READY** - купюроприёмник готов к работе (связь с устройством есть, стекер установлен, ошибок нет, однако это не означает включения приёма купюр)
BVSIMEVENT 13	**FULL** - переполнение стекера (рекомендуемая для отладки ошибка оборудования - может произойти даже на абсолютно исправном купюроприёмнике)
BVSIMEVENT 14	**DISPOSED** - снятие стекера (особое состояние, ошибкой не считается, но изменяет работу приложения **NDemia CashMatic KioskBrowser**)

`CONFIG PrinterSimulateNoPaper=<value>`

включение симуляции (имитации) отсутствия бумаги в принтере.
<value> - если ненулевое десятичное число, то включается симуляция отсутствия бумаги в принтере (для отладки клиентских приложений).
см. описание команды `CONFIG`
По умолчанию симуляция отсутствия бумаги отключена.
Симуляция отсутствия бумаги может выполняться в режиме вывода в файл.
Симуляция отсутствия бумаги не выполняется в следующих случаях:

- если отключена проверка статуса принтера;
- если задан режим "принтер не подключен" или если для печати выбран фискальный регистратор.

Обратите внимание: это значение записывается в реестр, поэтому использовать его надо с осторожностью - если включить и забыть об этом, то будет сложно разобраться, почему печать категорически не работает.

13 Пример платёжного интерфейса

При установке программного продукта **NDemia CashMatic KioskBrowser** в папке контента (**"\Program Files\NDemia\CashMatic\HTML"**) размещается пример прикладной программы: демонстрационный платёжный интерфейс на основе **NDemia CashMatic KioskBrowser**.
Пример показывает реализацию достаточно типичной платёжной системы, за исключением следующих обстоятельств:

- принятые платежи никуда не зачисляются, только печатается чек;
- пользователю-плательщику предлагается выбирать шаблон печати чека - это сделано с целью демонстрации различных вариантов формирования выходного документа.

В папке HTML при установке создаются следующие файлы и папки (демонстрационный интерфейс приёма платежей):

Имя	Комментарий
img/	папка, содержащая графический материал, использованный в примере
terminal.css	стили элементов (CSS), на которые ссылаются html-страницы примера
terminal.js	функции (javascript), используемые более чем на одной странице
start.html	стартовая страница (здесь показана необходимая инициализация и проверка общей готовности)
start.cmht	копия start.html, переименованная для удобства запуска из Проводника **Windows**
input.html	страница ввода данных о платеже (также здесь выбирается шаблон чека)
getcash.html	страница ввода наличных денег (здесь показана работа с купюроприёмником)
finish.html	завершающая страница (здесь выполняется печать чека)
failure.html	страница ошибки ("терминал временно не работает" - здесь показан механизм ожидания восстановления - при устранении проблемы терминал должен сам вернуться в рабочее состояние).

Примеры скриптов демонстрационного интерфейса приёма платежа содержат достаточно подробные комментарии, для ознакомления с работой программы стоит их просмотреть HTML-редактором в режиме исходного кода.