

Проект Тип оборудования
ServiceTerminal

Оглавление

1. ОСНОВНЫЕ ПОНЯТИЯ И ТЕРМИНЫ.....	3
2. ОПИСАНИЕ ТИПА.....	4
2.1. Наименование типа и русскоязычный синоним	4
2.2. Краткое описание функциональности типа	4
3. АЛГОРИТМЫ.....	4
3.1. Общие алгоритмы	4
Создание нового экземпляра оборудования	4
Использование экземпляра оборудования	5
Удаление экземпляра оборудования.....	6
3.2. Специфические алгоритмы	6
Настройка терминала услуг	6
Запрос параметров услуг	6
Авторизация услуги	6
4. НАСТРОЙКИ.....	6
4.1. Общие настройки	6
DefaultTimeOut.....	6
EventLogEnabled	7
4.2. Специфические настройки	7
5. МЕТОДЫ.....	7
5.1. Служебные методы.....	7
Init	7
InitProxyIC	7
Open	8
Close.....	8
GetDeviceInfo	9
5.2. Общие методы.....	9
GetSettings	10
SetSettings.....	11
ShowSettingsDlg	11
CheckHealth	12
Enable.....	12
Disable.....	12
5.3. Специфические методы.....	13
GetServiceCodes	13
GetServiceParams	13
AuthorizeService	15
CancelServiceAuthorization.....	15
GetAuthorizationStatus	16
Settlement.....	16
6. ФУНКЦИИ ОБРАТНОГО ВЫЗОВА.....	17
GetAppProperty.....	17

ErrorEvent	17
WriteLog	18

1. Основные понятия и термины

Термин	Описание
Подключаемое оборудование	Любое торговое или промышленное оборудование, использующееся клиентским приложением
Тип оборудования	Регламентированное документацией множество, объединяющее устройства с одинаковым функциональным назначением
Драйвер оборудования, Драйвер	Программный компонент с возможностью отдельной установки или обновления в системе и предоставляющий интерфейс управления и взаимодействия с устройством определенной модели. Предоставляемый драйвером интерфейс должен строго соответствовать описанию типа оборудования по данной технологии, но может иметь расширения (дополнительные методы и/или параметры) для выполнения функций, не регламентированных в документах описания системы
Модель, Модель оборудования	Множество устройств одного типа оборудования, одинаковых или близких по своим характеристикам, управляемых драйвером устройства. Модель оборудования определяется драйвером устройства. В случае, если один драйвер устройства позволяет единообразно управлять одной из нескольких сходных моделей физических устройств, для системы все эти устройства являются одной и той же моделью. Напротив, если для одного и того же физического устройства существует несколько разных драйверов, для системы они являются разными моделями.
Экземпляр оборудования	Совокупность файлов и данных о физическом устройстве и драйвере (идентификатор, наименование, список настроек и т. п.), хранимых и используемых для работы с данным устройством системой управления подключаемым оборудованием и приложением-клиентом.
Клиентское приложение	Любое приложение, использующее для управления подключаемым оборудованием описанные в документации по технологии интерфейсы и алгоритмы взаимодействия с драйверами устройств.
Менеджер оборудования	Часть клиентского приложения, в которой сосредоточены сервисные процедуры и регламентированные документацией интерфейсы взаимодействия с оборудованием.
Настройки устройства	Набор именованных значений, простых типов данных: String, Number, DateTime, Boolean, хранящийся вне клиентского приложения в файловом хранилище настроек. Для каждого определенного в системе устройства имеется свой набор настроек. Настройки устройства определяют параметры и режим функционирования устройства. Настройки доступны клиентскому приложению для чтения и записи с помощью специальных методов.
Уникальный идентификатор	В этом проекте – текстовое представление GUID. Идентификатор содержит 36 символов в верхнем регистре, не заключенных в фигурные скобки. Пример: 8A8910EC-78F5-41A5-9E18-7C28992CF580
Публикуемый метод	Один из определенных в описании интерфейса общих методов, либо методов типа. Все публикуемые методы имеют одинаковую структуру параметров. Входные и возвращаемые методом данные помещаются в два одномерных массива SafeArray
Функции обратного вызова	Регламентируемые данной документацией функции клиентского приложения, доступные для вызова через интерфейс IDispatch драйвером оборудования. Функции обратного вызова

Термин	Описание
	предназначены для информирования приложения о событиях или передачи данных.
Событие	Событие возникает как результат изменения состояния устройства. Событие может быть вызвано внешним воздействием, либо внутренними процессами в оборудовании. Для передачи событий приложению драйвер устройства использует функции обратного вызова клиентского приложения
Объект-абстракт	Объект драйвера устройства, не ассоциированный с экземпляром оборудования. Любой объект драйвера является объектом-абстрактом до выполнения метода Open. У объекта-абстракта допустим вызов только тех методов, которые не требуют взаимодействия с физическим устройством. Объект-абстракт не может использовать функции обратного вызова, поскольку не имеет собственного идентификатора в системе.
Объект драйвера	Объект драйвера устройства, созданный клиентским приложением. Для каждого физического устройства создается отдельный экземпляр объекта драйвера
Терминал услуг	Программно-аппаратный комплекс, умеющий производить авторизацию тех или иных услуг.

2. Описание типа

2.1. Наименование типа и русскоязычный синоним

Наименование типа англ.:	ServiceTerminal
Наименование типа рус.:	ТерминалУслуг
Синоним англ.:	Service terminal
Синоним рус.:	Терминал услуг

2.2. Краткое описание функциональности типа

Терминал услуг представляет собой программно-аппаратный комплекс для авторизации различных услуг. Услуги могут оплачиваться через сторонние платежные системы (к примеру Cyberplat, E-port и т.д.), или через другое ПО (бильярды, боулинги, сауны и т.д.).

Основная функциональность типа:

- Получить коды услуг
- Получить параметры услуги
- Авторизовать услугу
- Отменить авторизацию услуги
- Получить статус авторизации
- Сверка итогов

Обязательными методами являются:

- Получить коды услуг
- Получить параметры услуги
- Авторизовать услугу

3. Алгоритмы

3.1. Общие алгоритмы

В этой главе описаны общие для всех типов оборудования алгоритмы взаимодействия приложения с драйвером устройства.

Создание нового экземпляра оборудования

- Приложение создает объект драйвера устройства
- Следующим вызванным методом должен быть [Init](#), инициализирующий драйвер. При этом в параметрах метода драйверу передается ссылка на интерфейс IDispatch клиентского

приложения. Драйвер запоминает ссылку на интерфейс для обращения к функциям обратного вызова

- У созданного объекта вызывается метод [GetDeviceInfo](#) для получения массива [DeviceInfo](#), содержащего константную информацию о драйвере (закладывается разработчиком при реализации драйвера)
- Для получения значений настроек по умолчанию, у драйвера вызывается метод [GetSettings](#).
- Если драйвер имеет собственную форму для настройки оборудования (элемент [IsSettingsDlgExist](#) из структуры [DeviceInfo](#)), у него вызывается метод [ShowSettingsDlg](#) для её отображения. Если форма настроек отсутствует, приложение отображает свою универсальную форму, заполнив её значениями настроек по умолчанию
- Пользователь редактирует значения в форме настроек устройства и закрывает её
- Если пользователь отказался от сохранения настроек, нажав на форме кнопку «Отмена», считается, что он отказался от создания устройства. У драйвера вызывается метод [Close](#) и объект драйвера уничтожается. Процедура создания устройства прерывается.
- Если использовалась форма настройки приложения, то набор настроек передается в драйвер через метод [SetSettings](#) для проверки корректности их заполнения (валидации)
- Приложение генерирует уникальный идентификатор для нового экземпляра оборудования и создает соответствующий ему индивидуальный каталог в хранилище настроек
- У драйвера запрашивается набор его текущих (проверенных) настроек с помощью метода [GetSettings](#). Набор настроек данного экземпляра оборудования сохраняется в индивидуальном каталоге хранилища настроек в файле [Settings.xml](#) (см. документ «Установка и обновление системы управления подключаемым оборудованием»)
- У данного объекта драйвера вызывается метод [Close](#), после чего он выгружается из памяти приложения
- Клиентское приложение сохраняет идентификатор созданного устройства для последующего его использования (см. Использование экземпляра оборудования)

Использование экземпляра оборудования

В этом разделе описывается алгоритм работы с ранее созданным экземпляром оборудования.

- Приложение создает объект драйвера устройства. У созданного объекта вызывается метод [Init](#), инициализирующий драйвер. При этом в параметрах метода драйверу передается ссылка на интерфейс [IDispatch](#) клиентского приложения (используется драйвером для обращения к [функциям обратного вызова](#))
- Приложение вызывает метод драйвера [SetSettings](#), передавая в параметрах метода список значений настроек устройства. Драйвер применяет новые настройки.
- Приложение вызывает метод драйвера [Open](#), в параметрах которого передается идентификатор устройства. Данный идентификатор драйвер при обращениях к [функциям обратного вызова](#)
- Начальным состоянием любого устройства после выполнения метода [Open](#), является «Выключено». В этом состоянии допускается вызов следующих методов устройства: [GetDeviceInfo](#), [GetSettings](#), [SetSettings](#), [ShowSettingsDlg](#), [CheckHealth](#), [Enable](#), [Disable](#), [Close](#). Все остальные методы будут доступны после включения устройства.
- Перед началом реального использования оборудования приложение вызывает метод [Enable](#). При выполнении метода [Enable](#) драйвер должен выполнить все необходимые операции по подготовке к работе: подгрузить необходимые ему внешние библиотеки, открыть, захватить, либо заблокировать необходимые для работы ресурсы: коммуникационные порты, файлы и т.п. Если метод [Enable](#) завершается успешно, то устройство переходит в состояние «Включено».
- У оборудования в состоянии «Включено» клиентское приложение последовательно вызывает необходимые ему публикуемые методы драйвера устройства.
- Драйвер устройства может порождать различные события посредством вызова у приложения функций обратного вызова.
- По окончании работы с драйвером устройства, приложение должно отключить его, вызвав метод [Disable](#). При исполнении данного метода драйвер должен привести физическое устройство в исходное состояние, высвободить все захваченные им в процессе работы ресурсы (закрыть открытые им файлы, коммуникационные порты), выгрузить использовавшиеся сторонние библиотеки и т.п.
- Перед уничтожением объекта драйвера устройства, приложение вызывает метод-деструктор [Close](#). Драйвер очищает все свои служебные структуры данных, в том числе

очищает ссылку на интерфейс IDispatch клиентского приложения, переданную ему при вызове метода [Init](#).

Удаление экземпляра оборудования

Для удаления экземпляра подключаемого оборудования из системы, достаточно удалить индивидуальный каталог экземпляра оборудования из хранилища настроек (см. описание в документе «Программа установки системы управления подключаемым оборудованием.doc») и очистить в настройках клиентских приложений, использовавших данный экземпляр, все ссылки на его идентификатор.

3.2. Специфические алгоритмы

Настройка терминала услуг

Общие для устройства настройки могут варьироваться в зависимости от конкретной модели, поэтому они не вынесены в обязательные настройки, но на усмотрение разработчика они могут быть вынесены в специфическую форму настроек.

Для данного типа устройства актуальны также настройки специфические для каждой услуги, предполагается, что подобные настройки будут храниться во внешнем файле. А в самой форме настроек будет лишь указан путь до этого файла. Соответственно количество услуг и реквизиты услуг будут зависеть от этого файла.

Запрос параметров услуг

Для авторизации конкретной услуги необходимо сначала запросить у пользователя параметры данной услуги. Для того чтобы узнать какие параметры надо запрашивать у пользователя есть метод `GetServiceParams`, который вернет список необходимых параметров с масками ввода по коду услуги.

В самом простом случае эти параметры должны запрашиваться у пользователя последовательным вводом.

Полученные от пользователя реквизиты платежа лучше всего подтвердить, распечатав квитанцию с основными реквизитами и местом для подписи клиента.

Авторизация услуги

Запросив все нужные параметры услуги у пользователя, можно приступать непосредственно к операции авторизации методом `AuthorizeService`. При авторизации необходимо будет передать значения параметров в том же порядке, в котором мы получили массив параметров методом `GetServiceParams`, После авторизации приложение должно будет распечатать сลิป платежа (`SlipText`), а также напечатать информацию о платеже на фискальном чеке (параметр `PlainText`), если эти параметры возвращены.

4. Настройки

4.1. Общие настройки

В этом разделе описываются настройки общие для всех моделей данного типа оборудования. Данные настройки подлежат обязательной реализации в любом драйвере подключаемого оборудования.

DefaultTimeOut

Описание:

Значение таймаута выполнения метода по умолчанию в секундах. Если таймаут, преданный в параметрах какого-либо метода, равен нулю, драйвер использует в качестве таймаута значение данной настройки.

Тип	Допустимые значения	Значение по умолчанию	Представление
Number	10..65535	120	Таймаут выполнения метода по умолчанию, целое число (в сек.)

EventLogEnabled

Описание:

Данная настройка предназначена для включения режима записи диагностической информации в журнал событий, либо лог-файл. Используется в целях отладки и диагностики проблем с драйвером устройства. Если установлено значение True, драйвер вызывает функцию обратного вызова [WriteLog](#) в те моменты, когда это необходимо (определяется разработчиком). Значение по умолчанию для данной настройки – False

Тип	Допустимые значения	Значение по умолчанию	Представление
Boolean	True/False	False	Запись диагностической информации

4.2. Специфические настройки

Для типа оборудования «Терминал услуг» специфические настройки отсутствуют

5. Методы

5.1. Служебные методы

В данном разделе описаны методы, обеспечивающие взаимодействие клиентского приложения и драйвера на начальных и конечных этапах цикла использования оборудования. Данные методы подлежат обязательной реализации в любом драйвере.

Init

Синтаксис:

Init (ApplicationRef: COM-object, ErrorDescription: String): Boolean

Описание:

Метод-конструктор объекта драйвера. При выполнении данного метода происходит начальная инициализация драйвера. Метод вызывается единожды, сразу после создания объекта. Любые другие методы объекта драйвера могут быть вызваны только после успешного выполнения метода Init. Если метод вернул значение False, объект драйвера должен быть уничтожен, оставив систему в исходном состоянии. Если метод выполнен успешно, то с объектом драйвера можно работать как с объектом-абстрактом. Например получить список настроек по умолчанию, вызвав метод [GetSettings](#).

Параметры:

ApplicationRef: COM-object

Ссылка на интерфейс IDispatch приложения-клиента. Используя данную ссылку, драйвер может вызывать у приложения функции обратного вызова (после выполнения метода Open).

ErrorDescription: String

Если метод возвращает False, в данном параметре содержится описание ошибки.

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае. В случае, если метод вернул результат False, дальнейшая работа с драйвером невозможна и данный объект драйвера должен быть уничтожен

InitProxy1C

Синтаксис:

Init (LibType: Integer, AddinName: String, FileName: String, ModelID: String, ErrorDescription: String): Boolean

Описание:

Дополнительный метод-конструктор объекта драйвера 1C:Совместимо. При выполнении данного метода происходит начальная инициализация драйвера. Метод вызывается единожды, сразу после создания объекта и выполнения метода Init. Любые другие методы объекта драйвера 1C:Совместимо могут быть вызваны только после успешного выполнения метода InitProxy1C. Если метод вернул значение False, объект драйвера должен быть уничтожен, оставив систему в исходном состоянии. Если метод выполнен успешно, то с объектом драйвера

можно работать как с объектом-абстрактом. Например получить список настроек по умолчанию, вызвав метод [GetSettings](#).

Параметры:

LibType: Integer

Тип используемой библиотеки 1С:Совместимо:

- 1 – COM
- 2 – Native API

AddinName: String

ProgID 1С-компоненты для LibType 1. Или имя объекта из библиотеки для LibType 2..

FileName: String

имя файла библиотеки для LibType 2.

ModelID: String

уникальный идентификатор (GUID) модели.

ErrorDescription: String

Если метод возвращает False, в данном параметре содержится описание ошибки.

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае. В случае, если метод вернул результат False, дальнейшая работа с драйвером невозможна и данный объект драйвера должен быть уничтожен

Open

Синтаксис:

Open (DeviceID: String, ErrorDescription: String): Boolean

Описание:

Завершает инициализацию объекта драйвера и присваивает ему уникальный идентификатор экземпляра. В процессе выполнения данного метода, драйвер не должен захватывать какие-либо неразделяемые ресурсы (например, коммуникационные порты или файлы), и не должен выполнять никаких реальных взаимодействий с устройством. Драйвер проверяет наличие необходимых ему для работы ресурсов (файлов библиотек, их версий и т.п.) и возвращает Ложь, если проверки обнаружили невозможность полноценного функционирования драйвера. После успешного выполнения метода допускается вызов следующих методов драйвера: [ShowSettingsDlg](#), [CheckHealth](#), [Enable](#), [Disable](#).. Все остальные методы будут доступны после включения устройства (метод [Enable](#)).

После выполнения метода драйвер может использовать [функции обратного вызова](#).

Параметры:

DeviceID: String (GUID)

Уникальный идентификатор экземпляра оборудования. Идентификатор необходим при использовании функций обратного вызова.

ErrorDescription: String

Если метод вернул False, данный параметр содержит описание ошибки.

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае. В случае, если метод вернул результат False, дальнейшая работа с драйвером невозможна. Необходимо вызвать метод [Close](#) и уничтожить объект драйвера.

Close

Синтаксис:

Close (): Boolean

Описание:

Метод-деструктор драйвера. Метод должен вызываться непосредственно перед уничтожением объекта драйвера. При выполнении данного метода, драйвер устройства должен освободить все захваченные ресурсы, и очистить ссылку на интерфейс IDispatch приложения, переданную в параметрах метода [Init](#). Никакие другие методы драйвера не могут быть вызваны после вызова [Close](#)

Параметры:

Нет

Возвращаемое значение:

True

GetDeviceInfo

Синтаксис:

GetDeviceInfo (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Метод предназначен для получения статической информации о модели устройства. Допустимо вызывать данный метод в любой момент, даже у неинициализированного объекта (то есть до метода [Init](#))

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: SafeArray

При успешном выполнении метода данный параметр содержит массив SafeArray с информацией о модели устройства. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

Таймаут ожидания результата выполнения метода. (см. [TimeOut](#))

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	TypeName	String	Наименование типа оборудования
1	ModelID	String	GUID модели оборудования
2	ModelName	String	Наименование модели
3	MajorVersion	Number	Мажор-версия системы (контроль совместимости на строгое равенство)
4	MinorVersion	Number	Минор-версия системы (совместимость снизу-вверх)
5	BuildVersion	Number	Текущая версия драйвера (билд), никак не контролируется системой
6	ProcessorName	String	Программный идентификатор драйвера. Используется для создания объекта драйвера
7	ProcessorType	Number	Тип драйвера: 1 = COM объект Win32, 2 = COM объект Win64, 3 = Native API Win32, 4 = Native API Win64, 5 = Native API Linux32, 6 = Native API Linux64
8	IsSettingsDlgExist	Boolean	Наличие собственной формы настроек (см. описание метода ShowSettingsDlg)
9	Description	String	Подробное описание модели
10	DeveloperName	String	Разработчик
11	CI_email	String	Контактная информация
12	CI_www	String	Контактная информация
13	CI_Phones	String	Контактная информация
14	CI_address	String	Контактная информация

5.2. Общие методы

Перечисленные в данном разделе методы являются обязательными для реализации в любом драйвере любого типа оборудования, использующего данную технологию. Совокупность общих и специфических методов называется «публикуемыми» методами. Публикуемые методы доступны для вызова из любого места приложения, в то время как служебные методы должен использовать только менеджер оборудования и строго по описанным алгоритмам взаимодействия. Вызов любого публикуемого метода возможен только у полностью инициализированного объекта

драйвера (см. описание метода [Open](#)) за исключением методов [GetSettings](#), [GetDeviceInfo](#) и [ShowSettingsDlg](#), которые допустимо вызывать у объекта-абстракта. Входные и выходные параметры публикуемых методов всегда помещаются в одномерные массивы SafeArray. Все публикуемые методы всегда имеют три параметра:

InputParameters: SafeArray

Одномерный массив SafeArray с входными параметрами метода, либо Неопределено (Undefined) когда входные параметры отсутствуют.

ResultData: SafeArray

Одномерный массив SafeArray с результатами выполнения метода, либо Неопределенно (Undefined), если возвращаемые параметры отсутствуют. В случае, когда метод возвращает False, в данном параметре содержится массив, содержащий код и описание ошибки (см. описание массива [ErrorInfo](#))

TimeOut: Number

Определяет максимальное время ожидания выполнения метода в секундах. Имеются два зарезервированных предопределенных значения:

0 – время исполнения метода не задано. Драйвер должен использовать значение настройки [DefaultTimeOut](#).

-1 – время исполнения метода не ограничено. Таймаут может быть только целым числом

Массив ErrorInfo:

Индекс в SafeArray	Имя	Тип	Описание
0	ErrorCode	Number	Регламентированный код ошибки (см. документ «Коды ошибок и их диапазоны.doc»)
1	ErrorDescription	String	Описание ошибки оборудования
2	ExtData	SafeArray / String / Number / DateTime / Boolean	Дополнительные данные или Undefined. Тип параметра определяется типом оборудования и вызываемым методом. Необязательный

GetSettings

Синтаксис:

GetSettings (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Получение списка значений текущих настроек устройства. Если метод вызывается до первого вызова метода SetSettings, возвращаются значения настроек устройства по умолчанию, заданные разработчиком драйвера.

Параметры:

InputParameters: Undefined

Не используется. При вызове необходимо передавать значение Undefined.

ResultData: SafeArray

Массив значений настроек [Settings](#). В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

Таймаут ожидания результата выполнения метода. (см. [TimeOut](#))

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае.

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	Settings	SafeArray	Массив значений настроек устройства

Массив Settings

Двумерный массив значений настроек устройства. Индекс старшего измерения (строки массива) изменяется в диапазоне от 0 до N (N = количество настроек - 1). Индекс младшего измерения (колонки массива) изменяется в диапазоне от 0 до 6

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	ID	String	Идентификатор настройки. Идентификатор должен начинаться с буквы. Допустимо использование только латинских символов.
(0..N, 1)	Value	Number / String / DateTime / Boolean	Значение настройки
(0..N, 2)	Type	String	Строковое представление типа значения настройки. Допустимые варианты: «Number» «String» «DateTime» «Boolean»
(0..N, 3)	Default	Number / String / DateTime / Boolean	Значение настройки по умолчанию
(0..N, 4)	ReadOnly	Boolean	Если равно True, то пользователям запрещается изменять значение данной настройки
(0..N, 5)	Name	String	Пользовательское представление настройки
(0..N, 6)	Description	String	Текстовое описание настройки, ее возможных значений и их воздействия на поведение драйвера

SetSettings

Синтаксис:

SetSettings (InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Метод предназначен для установки новых значений настроек устройства. Допустимо вызывать данный метод до вызова метода [Open](#). Метод не должен вызываться приложением, если устройство находится в состоянии «Включено» (см. метод [Enable](#)).

Параметры:

InputParameters: SafeArray

Массив значений настроек [Settings](#).

ResultData: SafeArray либо Undefined

Возвращаемые параметры отсутствуют. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

Таймаут ожидания результата выполнения метода. (см. [TimeOut](#))

Возвращаемое значение:

True – метод выполнен успешно. False – в противном случае

Массив InputParameters:

Индекс SafeArray	Имя	Тип	Описание
0	Settings	SafeArray	Массив значений настроек устройства

ShowSettingsDlg

Синтаксис:

ShowSettingsDlg (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

При выполнении данного метода, драйвер отображает в модальном режиме форму для интерактивного изменения настроек устройства. Допускается отсутствие собственной формы настроек при реализации драйвера. В этом случае, метод возвращает False и ошибку с кодом 1003 (см. [ErrorInfo](#)), а клиентское приложение отображает собственную универсальную форму для редактирования настроек устройства.

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: SafeArray

При отсутствии у драйвера формы настроек, возвращается [ErrorInfo](#) с кодом ошибки 1003. Если пользователь отказался от внесения изменений (нажал кнопку «Отмена») возвращается [ErrorInfo](#) с кодом ошибки 201. В случае, если пользователь нажал кнопку «ОК» и настройки были успешно применены, содержит значение Undefined

TimeOut: Number

Таймаут ожидания результата выполнения метода. Значение данного параметра игнорируется

Возвращаемое значение:

True – метод выполнен успешно, настройки изменены и применены новые значения настроек.

False – изменения настроек не произошло.

CheckHealth

Синтаксис:

CheckHealth (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Метод используется для проверки работоспособности устройства. Вызов данного метода может приводить к исполнению длительных операций, поэтому в клиентском приложении не рекомендуется использовать данный метод в обычном цикле использования оборудования.

Технология никак не регламентирует действия драйвера при выполнении данного метода.

Состав производимых проверок определяется разработчиком драйвера.

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: Undefined либо SafeArray

Возвращаемые параметры отсутствуют. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

Таймаут ожидания результата выполнения метода. Значение данного параметра игнорируется

Возвращаемое значение:

True – устройство исправно и работоспособно. False – в противном случае

Enable

Синтаксис:

Enable (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Включение устройства. Метод должен быть вызван перед началом использования оборудования, до вызова любого специфического метода. При выполнении данного метода драйвер проводит все возможные проверки на готовность устройства к работе и только в случае их успешного завершения, возвращает результат True. Если драйверу требуется для работы монополюсный захват каких-либо ресурсов (например, коммуникационный порт), он должен выполнить его при выполнении данного метода.

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ErrorInfo: SafeArray либо Undefined

Возвращаемые параметры отсутствуют. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

Таймаут ожидания результата выполнения метода. (см. [TimeOut](#))

Возвращаемое значение:

True – устройство включено и готово к работе. False – в противном случае.

Disable

Синтаксис:

Disable (InputParameters: Undefined, ResultData: SafeArray, TimeOut: Number): Boolean

Описание:

Выключение устройства. Если драйвером были монополюсно захвачены какие-либо ресурсы (например, коммуникационный порт), он должен освободить их при выполнении данного метода.

Параметры:*InputParameters: Undefined*

Не используется. Необходимо передавать пустое значение.

*ErrorInfo: SafeArray*Возвращаемые параметры отсутствуют. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)*TimeOut: Number*Таймаут ожидания результата выполнения метода. (см. [TimeOut](#))**Возвращаемое значение:**

True – устройство успешно выключено. False – в противном случае

5.3. Специфические методы

Перечисленные в данном разделе методы являются специфическими методами типа оборудования. Любой из этих методов может быть вызван только у включенного устройства (см. [Enable](#)).

GetServiceCodes**Синтаксис:***GetServiceCodes(InputParameters: Undefined, ResultData: SafeArray, TimeOut: Numbers): Boolean***Описание:**

Получает список услуг, предоставляемых терминалом услуг. Список услуг представляет массив с Кодом услуги и представлением услуги.

Параметры:*InputParameters: Undefined*

Не используется. Необходимо передавать пустое значение.

*ResultData: SafeArray*Содержит список предоставляемых услуг. В случае ошибки – стандартный массив [ErrorInfo](#).*TimeOut: Numbers*Таймаут исполнения команды в секундах. См. [TimeOut](#)**Возвращаемое значение:**

True – функция выполнена успешно. False – в противном случае.

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	ServiceCode	String (10)	Идентификатор услуги
(0..N, 1)	ServiceName	String	Представление услуги

GetServiceParams**Синтаксис:***GetServiceParams(InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Numbers): Boolean***Описание:**

Получает список параметров по идентификатору услуги, которые необходимо будет запросить у пользователя.

Параметры:*InputParameters: Undefined*

Не используется. Необходимо передавать пустое значение.

*ResultData: SafeArray*Содержит список предоставляемых услуг. В случае ошибки – стандартный массив [ErrorInfo](#).*TimeOut: Numbers*Таймаут исполнения команды в секундах. См. [TimeOut](#)**Возвращаемое значение:**

True – функция выполнена успешно. False – в противном случае.

Массив InputParameters:

Индекс SafeArray	Имя	Тип	Описание
---------------------	-----	-----	----------

0	ServiceCode	String(10)	Идентификатор услуги
---	-------------	------------	----------------------

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	ParameterName	String	Представление параметра, значение которого необходимо будет запросить у пользователя
(0..N, 1)	DefaultData	String	Строка со значением по умолчанию или пустая строка, если такового нет. При открытии формы диалога, поле ввода должно быть заполнено значением данного параметра.
(0..N, 2)	Description	String	Детальное описание запрашиваемой информации и порядка ее ввода. Переданный в данном параметре текст, приложение может отобразить рядом с полем ввода или в подсказке к нему. Необязательный параметр.
(0..N, 3)	Mask	String	<p>Маска для поля ввода. Необязательный параметр. В строке маски допустимо использование следующих специальных символов:</p> <ul style="list-style-type: none"> ! - любой введенный символ преобразуется в верхний регистр; 9 - допустимо ввести произвольный символ цифры; # - допустимо ввести произвольный символ цифры или - (знак минус) или + (знак плюс) или пробел; N - допустимо ввести любые алфавитно-цифровые символы (буквы или цифры); U - допустимо ввести любые алфавитно-цифровые символы (буквы или цифры) и любой введенный символ преобразуется в верхний регистр; X (латинского алфавита) - допустимо ввести произвольный символ; ^ - не допустимо вводить этот символ интерактивно пользователем, он может устанавливаться только из языка; h - допустим ввод символов обозначения шестнадцатеричных цифр; @ – допустимо ввести любые алфавитно-цифровые символы (буквы или цифры) в верхнем регистре или пробел. <p>При помещении значения из поля ввода с маской в текстовый реквизит, связанный с этим полем ввода, происходит следующее преобразование: на тех позициях, где в маске стоит символ "@", а в строке пробел – пробел удаляется. Если в маске из специальных символов используются только символы "@", то все символы текста, соответствующие символам маски, не являющимися специальными символами, удаляются после последнего непустого блока из символов "@". Например, при маске "@@.@@.@@" текст "41. 2. ." преобразуется в "41.2".</p> <p>Для того, чтобы использовать в маске один из специальных символов, нужно использовать перед ним символ "\".</p> <p>Допускается указание нескольких масок в</p>

			одном параметре. Маски разделяются символом ";". В этом случае использоваться будет та маска, к которой подходит введенный текст.
--	--	--	---

AuthorizeService

Синтаксис:

AuthorizeService(InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Numbers): Boolean

Описание:

Производит авторизацию услуги по идентификатору услуги. В зависимости от вида услуги могут использоваться параметры: количество услуг, сумма оплаты и итоговая сумма. В массиве параметров передаются параметры услуги в том порядке, в котором они были получены методом *GetServiceParams*. В возвратных параметрах передается номер проведенной транзакции, для последующей идентификации операции; текст, который необходимо напечатать на фискальном чеке; массив параметров, которые использовались при проведении авторизации, на тот случай, если приложение захочет хранить у себя параметры авторизации.

Параметры:

InputParameters: SafeArray

Массив входных параметров

ResultData: SafeArray

Массив возвращаемых данных. В случае ошибки – стандартный массив [ErrorInfo](#).

TimeOut: Numbers

Таймаут исполнения команды в секундах. См. [TimeOut](#)

Возвращаемое значение:

True – функция выполнена успешно. False – в противном случае.

Массив InputParameters:

Индекс SafeArray	Имя	Тип	Описание
0	ServiceCode	String(10)	Идентификатор услуги
1	Count	Number	Количество услуг, на которое производится авторизация
2	Sum	Number	Сумма оплаты услуги платежной системы
3	TotalSum	Number	Сумма оплаты, полученная с клиента (с учетом комиссии сверху)
4	Parameters	SafeArray	Массив параметров оплаты

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	ReferenceNumber	String	Номер проведенной транзакции
1	PlainText	String	Текст для фискального чека
2	SlipText	String	Текст слипа

Массив Parameters:

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	ParameterName	String	Представление параметра
(0..N, 1)	ParameterValue	String	Значение параметра

CancelServiceAuthorization

Синтаксис:

CancelServiceAuthorization(InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Numbers): Boolean

Описание:

Производит отмену авторизацию услуги по номеру проведенной транзакции.

Параметры:

InputParameters: SafeArray

Массив входных параметров

ResultData: SafeArray

Массив возвращаемых данных. В случае ошибки – стандартный массив [ErrorInfo](#).

TimeOut: Numbers

Таймаут исполнения команды в секундах. См. [TimeOut](#)

Возвращаемое значение:

True – функция выполнена успешно. False – в противном случае.

Массив InputParameters:

Индекс SafeArray	Имя	Тип	Описание
0	ReferenceNumber	String	Номер отменяемой транзакции
1	Count	Number	Количество услуг, на которое производится отмена авторизации
2	Sum	Number	Сумма оплаты услуги платежной системы, на которую производится отменяя
3	TotalSum	Number	Сумма оплаты, возвращаемая клиенту (с учетом комиссии сверху)

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	PlainText	String	Текст для фискального чека
1	SlipText	String	Текст слипа

GetAuthorizationStatus

Синтаксис:

GetAuthorizationStatus(InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Numbers): Boolean

Описание:

Функция производит проверку статуса платежа. Для проверки статуса платежа в параметре ReferenceNumber передается код транзакции, в случае успешного выполнения операции в возвращаемом параметре AnswerCode содержится код ответа, а в параметре AnswerMessage содержится текст ответа.

Параметры:

InputParameters: SafeArray

Массив входных параметров.

ResultData: SafeArray

Массив возвращаемых данных. В случае ошибки – стандартный массив [ErrorInfo](#).

TimeOut: Numbers

Таймаут исполнения команды в секундах. См. [TimeOut](#)

Возвращаемое значение:

True – функция выполнена успешно. False – в противном случае.

Массив InputParameters:

Индекс SafeArray	Имя	Тип	Описание
0	ReferenceNumber	String	Номер транзакции

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	AnswerCode	Number	Код ответа (если код ответа = 0, то авторизация прошла успешно)
1	AnswerMessage	String	Текст ответа

Settlement

Синтаксис:

Settlement(InputParameters: SafeArray, ResultData: SafeArray, TimeOut: Numbers): Boolean

Описание:

Функция сверки итогов дня (закрытие смены платежного терминала). Функция возвращает итоги по количеству проведенных платежей, суммам полученным с клиента и суммам переданным в платежный терминал за текущую смену. В случае, если конкретная модель термина услуг не поддерживает сверку (не нуждается в ней), метод возвращает True не выполняя никаких действий.

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: SafeArray

Массив возвращаемых данных. В случае ошибки – стандартный массив [ErrorInfo](#).

TimeOut: Numbers

Таймаут исполнения команды в секундах. См. [TimeOut](#)

Возвращаемое значение:

True – функция выполнена успешно. False – в противном случае.

Массив ResultData:

Индекс SafeArray	Имя	Тип	Описание
0	AuthorizationCount	Number	Количество авторизаций за смену
1	Sum	Number	Сумма, переданная платежной системе
2	TotalSum	Number	Сумма, полученная от клиента (с учетом комиссии сверху)

6. Функции обратного вызова

В этом разделе описаны функции, подлежащие обязательной реализации в менеджере управления оборудованием любого клиентского приложения. Все функции предназначены для вызова их драйверами устройств.

При вызове метода [Init](#), в его параметрах передается ссылка на интерфейс IDispatch приложения-клиента. Используя эту ссылку, драйвер может вызывать функции обратного вызова, реализованные в приложении.

Для удаленно используемого оборудования вызовы функций будут перенаправляться удаленным клиентским приложениям, через сетевой транспорт.

GetAppProperty

Синтаксис:

GetAppProperty (SourceID: String, Name: String, Value: String): Boolean

Описание:

Функция предназначена для получения свойств системы и приложения-клиента.

Параметры:

SourceID: String (GUID)

Идентификатор устройства, сгенерировавшего событие

Name: String

Наименование требуемого свойства. Допустимые значения:

- «ModelsFolder» - Путь к каталогу моделей оборудования
- «DevicesFolder» - Путь к каталогу экземпляров оборудования
- «Language» - Язык локализации приложения

Value: String

В этом параметре возвращается значение свойства. Для свойства «Language» возвращается двухсимвольный идентификатор языка («ru», «en», «ua» и т. д.)

Возвращаемое значение:

True – функция выполнена успешно, указанное свойство найдено. False – в противном случае

ErrorEvent

Синтаксис:

ErrorEvent (SourceID: String, ErrorName: String, Description: String, ExtData: SafeArray): Boolean

Описание:

Функция предназначена для передачи приложению информации о возникновении ошибочного состояния или аварийной ситуации в устройстве. Недопустим вызов данной функции во время выполнения драйвером устройства какого-либо метода. Клиентское приложение при вызове данной функции должно любым доступным образом сообщить пользователю о возникшей ошибке.

Параметры:

SourceID: String (GUID)

Идентификатор устройства, сгенерировавшего событие

ErrorMessage: String

Строка с наименованием типа ошибки. Типы ошибок регламентируются на уровне типов оборудования

Description: String

Строка с кратким описанием возникшей ошибки или аварийной ситуации в устройстве

ExtData: SafeArray либо Undefined

Массив, содержащий дополнительные данные об ошибке. Структура массива определяется типом ошибки и типом оборудования. Необязательный параметр.

Возвращаемое значение:

True – вызов обработан успешно. False – в противном случае

WriteLog

Синтаксис:

WriteLog (SourceID: String, Event: String, Text: String, Type: Number): Boolean

Описание:

Функция предназначена для передачи диагностической и отладочной информации приложению-клиенту. Приложение должно сохранять данную информацию в журнале (лог-файле). Данную функцию можно вызывать в любой момент – на усмотрение разработчика драйвера.

Рекомендуется использовать её для диагностики проблем и ошибок при работе с устройством.

Если настройка драйвера [EventLogEnabled](#) имеет значение False, функция вызываться не должна.

Параметры:

SourceID: String (GUID)

Идентификатор устройства, сгенерировавшего событие

Event: String

Наименование события. Произвольная строка. По наименованию событий можно фильтровать/группировать записи в журнале.

Text: String

Описание события. Произвольная строка

Type: Number

Тип события. Допустимые значения: 0 – Информация; 1 – Ошибка; 2 - Отладка

Возвращаемое значение:

True – функция выполнена успешно, информация записана приложением в журнал. False – в противном случае