

Проект Тип оборудования Card Reader

Оглавление

1. ОСНОВНЫЕ ПОНЯТИЯ И ТЕРМИНЫ	4
2. ОПИСАНИЕ СТАНДАРТА	5
2.1. Наименование типа и синоним	5
2.2. Краткое описание функциональности типа	5
3. АЛГОРИТМЫ	5
3.1. Общие алгоритмы	5
Создание нового экземпляра оборудования	5
Использование экземпляра оборудования	6
Удаление экземпляра оборудования	6
3.2. Специфические алгоритмы	7
4. НАСТРОЙКИ	7
4.1. Общие настройки	7
DefaultTimeOut	7
EventLogEnabled	7
4.2. Специфические настройки	8
5. МЕТОДЫ	8
5.1. Служебные методы	8
Init	8
InitProxy1C	8
Open	9
Close	9
GetDeviceInfo	9
Описание:	9
5.2. Общие методы	10
GetSettings	11
SetSettings	12
ShowSettingsDlg	12
CheckHealth	13
Enable	13
Disable	13
5.3. Специфические методы	14
EnableDataEvents	14
GetDataCount	14
GetTracksData	15
WriteTracks	15
6. ФУНКЦИИ ОБРАТНОГО ВЫЗОВА	16
GetAppProperty	16
DataEvent	16
ErrorEvent	17

WriteLog.....17

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 предназначены для информирования приложения о событиях или передачи данных.

Термин	Описание
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
Считыватель карт, Card Reader	Устройство для считывания информации с пластиковой карты (магнитной, штрихкодовой, RFID).
Дорожка/Сектор	Блок данных карты. Данные разбиты по блокам по своему функциональному назначению. На одной из дорожек (далее – основная дорожка) записан идентификатор карты. У магнитных карт обычно имеется не более 4-х дорожек. У RFID карт может быть произвольное количество секторов

2. Описание стандарта

2.1. Наименование типа и синоним

Наименование типа англ.:	CardReader
Наименование типа рус:	СчитывательКарт
Синоним англ.:	Card Reader
Синоним рус:	Считыватель Карт

2.2. Краткое описание функциональности типа

Основное предназначение типа – уведомление приложения о считывании карты. В данном случае, под картой может подразумеваться любая идентификационная карта – с магнитной полосой, штрихкодом или RFID – карта. Данные карты разделены на блоки – дорожки или сектора. Каждый блок может содержать произвольную текстовую информацию. Драйвер должен ожидать чтения карты устройством. При прочтении, драйвер должен передать данные карты приложению-клиенту. Если приложение не готово к приему данных, драйвер может сохранить данные карты в своем буфере для последующей передачи.

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

После считывания карты, драйвер вызывает метод менеджера оборудования DataEvent, если это не запрещено текущим режимом работы. Иначе – данные помещаются в буфер драйвера. Как только приложение меняет режим работы на режим, позволяющий передавать данные, все накопленные данные из буфера драйвера передаются приложению в порядке их поступления, путем последовательного вызова функции обратного вызова DataEvent.

Также возможен режим работы драйвера, в котором приложение получает данные только тогда, когда это ему необходимо. В этом случае, вызов функции DataEvent не происходит вовсе, все полученные от сканера данные помещаются в очередь. Приложение вызывает метод GetTracksData тогда, когда ему необходимо получить из очереди данные очередной считанной карты.

Особый режим работы – режим автоотключения считывателя. Работая в данном режиме, считыватель выключается после считывания одной карты. Дальнейшее считывание карт считывателем становится невозможным (если считыватель поддерживает аппаратное отключение). После того, как данные карты будут обработаны приложением, оно включит считыватель, используя метод EnableDataEvents.

Помимо этого, некоторые модели считывателей могут производить запись данных на карту, при помощи метода WriteTracks.

Считываться и записываться может как одна дорожка, так и массив дорожек. В первом случае, настройки драйвера должны определять, какая дорожка является основной для записи, и какая – для чтения. Как правило, основная дорожка содержит собственно идентификатор карты. Для магнитных карт обычно это – вторая дорожка.

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

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

InitProху1C

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 InitProху1C. 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 1C:Совместимо:

- 1 – COM
- 2 – Native API

AddinName: String

ProgID 1C-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 <code>SafeArray</code>	Имя	Тип	Описание
0	<code>ErrorCode</code>	Number	Регламентированный код ошибки (см. документ «Коды ошибок и их диапазоны.doc»)
1	<code>ErrorDescription</code>	String	Описание ошибки оборудования
2	<code>ExtData</code>	<code>SafeArray</code> / String / Number / DateTime / Boolean	Дополнительные данные или <code>Undefined</code> . 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 <code>SafeArray</code>	Имя	Тип	Описание
0	Settings	<code>SafeArray</code>	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 <code>SafeArray</code>	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](#)).

EnableDataEvents

Синтаксис:

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

Описание:

Метод управляет режимом передачи данных приложению. Если включен режим передачи событий – драйвер передает данные приложению сразу после их поступления от считывателя. Если данный режим отключен, драйвер ставит данные в очередь и приложение получает их в тот момент, когда ему это необходимо (см. метод [GetTrackData](#)).

Параметры:

InputParameters: SafeArray

Массив входных параметров. Содержит один элемент.

ResultData: Undefined

Не используется. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

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

Возвращаемое значение:

True – метод выполнен успешно. В противном случае – False

Массив InputParameters

Индекс SafeArray	Имя	Тип	Описание
0	Value	Boolean	Если значение равно True, то вызов метода DataEvent происходит при получении данных от считывателя, либо сразу после выполнения EnableDataEvent , если на момент вызова в буфере драйвера находились еще не отправленные в приложение данные. При установке значения False, драйвер перестает вызывать функцию DataEvent , и все получаемые данные сохраняет в своем буфере. Клиентское приложение может получить их при помощи вызова метода GetTracksData .

GetDataCount

Синтаксис:

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

Описание:

Используется приложением, для определения количества записей данных в очереди драйвера

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: SafeArray

Массив возвращаемых данных. Содержит один элемент.

TimeOut: Number

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

Возвращаемое значение:

True – метод выполнен успешно. В противном случае – False

Массив ResultData

Индекс SafeArray	Имя	Тип	Описание
0	Value	Number	Содержит количество записей данных, полученных

			от устройства, находящихся в очереди драйвера.
--	--	--	------------------------------------------------

GetTracksData

Синтаксис:

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

Описание:

Возвращает данные последней карты из очереди драйвера. Данные дорожек должны передаваться без префикса, постфикса и спец. символов, добавляемых считывателем.

Параметры:

InputParameters: Undefined

Не используется. Необходимо передавать пустое значение.

ResultData: SafeArray

Массив возвращаемых данных. Количество элементов равно значению настройки TracksCount.

TimeOut: Number

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

Возвращаемое значение:

True – метод выполнен успешно. В противном случае – False

Массив ResultData

Индекс SafeArray	Имя	Тип	Описание
0	MainTrackData	Строка	Содержит данные основной дорожки/сектора карты. Какая из дорожек считается основной, определяется настройками драйвера. Для магнитных карт, это, как правило, вторая дорожка, на которой записан код, по которому приложение идентифицирует данную карту.
1	TracksData	SafeArray	Массив с данными дорожек или Undefined, если считывается только основная дорожка

Массив TracksData

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	TrackNum	Number	Номер дорожки/сектора карты
(0..N, 1)	TrackData	String	Данные дорожки/сектора карты.

WriteTracks

Синтаксис:

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

Описание:

Иницирует запись дорожек/секторов на карту. Необязательный.

Параметры:

InputParameters: Undefined

Массив входных параметров.

ResultData: SafeArray

Не используется. В случае ошибки выполнения метода, параметр содержит массив [ErrorInfo](#)

TimeOut: Number

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

Возвращаемое значение:

True – метод выполнен успешно. В противном случае – False

Массив InputParameters

Индекс SafeArray	Имя	Тип	Описание
0	MainTrackData	String	Данные для записи основной дорожки/сектора карты. Какая из дорожек считается основной для записи, определяется настройками драйвера. Для магнитных

			карт, это, как правило, вторая дорожка, на которой записан код, по которому приложение идентифицирует данную карту.
1	TracksData	SafeArray	Массив с данными дорожек для записи или Undefined, если записывается только основная дорожка

Массив TracksData

Индекс SafeArray	Имя	Тип	Описание
(0..N, 0)	TrackNum	Number	Номер записываемой дорожки/сектора
(0..N, 1)	TrackData	String	Данные для записи дорожки/сектора карты.

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 типа оборудования "Считыватель" не предполагается использование таких функций обратного вызова как MessageDlg, InputDataDlg и TaskState

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

DataEvent

Синтаксис:

DataEvent (SourceID: String, Event: String, Data: String, ExtData: SafeArray): Boolean

Описание:

Функция предназначена для информирования приложения драйвером устройства о событии и передачи данных. Для типа оборудования «CardReader», данное событие генерируется драйвером для передачи приложению данных основной дорожки/сектора считанной карты и возникает только в режиме получения данных по событию (см. EnableDataEvents). Если событие распознано и обработано приложением, функция возвращает True, иначе возвращается False.

Параметры:

SourceID: String

Идентификатор устройства сгенерировавшее событие

Event: String

Строка с наименованием типа события. Драйвер оборудования может генерировать события разных типов.

Для типа «Считыватель» предусмотрено одно значение – «TracksData». Событие инициируется драйвером при считывании драйвером карты, когда настройки считывателя этого не запрещают

Data: String

Строка, содержащая краткое представление данных.

Для типа оборудования «CardReader», в данном параметре содержится строка, содержащая данные основной дорожки/сектора без префикса, постфикса и спец. символов, добавляемых считывателем. Какая из дорожек считается основной, определяется настройками драйвера. Для магнитных карт, это, как правило, вторая дорожка, на которой записан код, по которому приложение идентифицирует данную карту.

ExtData: SafeArray либо Undefined

Массив, содержащий расширенные (полные) данные.

Для типа оборудования «CardReader» - массив, структура которого идентична структуре [TracksData](#). Возвращает данные всех предназначенных для чтения дорожек/секторов карты.

Возвращаемое значение:

True – функция выполнена успешно, данные события доставлены в клиентское приложение.

False – событие не было обработано клиентским приложением

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

Параметры:

SourceID: String (GUID)

Идентификатор устройства сгенерировавшего событие

ErrorName: String

Строка с наименованием типа ошибки. Драйвер оборудования может генерировать ошибки разных типов.

Description: String

Строка с кратким описанием возникшей ошибки или аварийной ситуации в устройстве

ExtData: SafeArray

Массив, содержащий дополнительные данные об ошибке. Необязательный.

Возвращаемое значение:

True – функция выполнена успешно, ошибка обработана. 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