DUDE

(ver.1.1.0.3)

Tutorial and overview

Disclaimer

Information in this publication is subject to change without notice and does not represent a commitment. However, as product improvements become available, we will make every effort to provide updated information for the products described in this publication.

History

Changes to the original guide are listed below:

Version	Date	Description
1.1.0.3	10/2024	✔ Added support for: • **"FP-950MX"**
1.1.0.2	10/2021	✔ Added support for: • **"WP-50MX"** • **"DP-25MX** • **"DP-150MX**
1.1.0.1	09/2021	✔ Added support for: • **"SK1-21"**
1.1.0.0	05/2020	✔ Added support for: • **"DP-05L"**
1.0.0.2	06/2018	✔ Added support for: • **"FP-650V"** • **"FP-700V"** • **"FP-800V"** • **"FP-2000"**
1.0.0.1	06/2018	✔ Added support for: • **"FMP-350"** • **"FP-650"**
1.0.0.0	06/2018	Small bug fixes. Documentation update.
1.0.0.0	01/2018	Initital release. ✔ Added support for models: • **"DP-05"** • **"DP-25"** • **"DP-35"** • **"DP-150"** • **"WP-50"** • **"WP-500"** • **"FP-700"** • **"FP-800"**

Introduction

This document provides an information about the Datecs COM Server "DUDE" which is a driver for all fiscal devices (production of Datecs) in Romania after the beginning of 2018. The COM server enables easy integration and usage of the fiscal devices of Datecs under OS Windows.

If you are not familiar with the COM technology – in short, the COM server is an object that provides services to clients. These services are in the form of COM interface implementations that can be called by any client that is able to get a pointer to one of the interfaces on the server object. There are two main types of servers, in-process and out-of-process. DUDE is an out-of-process COM server. The essence of COM is a language-neutral way of implementing objects that can be used in environments different from the one in which they were created, even across machine boundaries.

DUDE allows you, as a programmer, to manage the fiscal devices under different types of the connections – RS232, USB or via TCP/IP (depended by model).

Languages

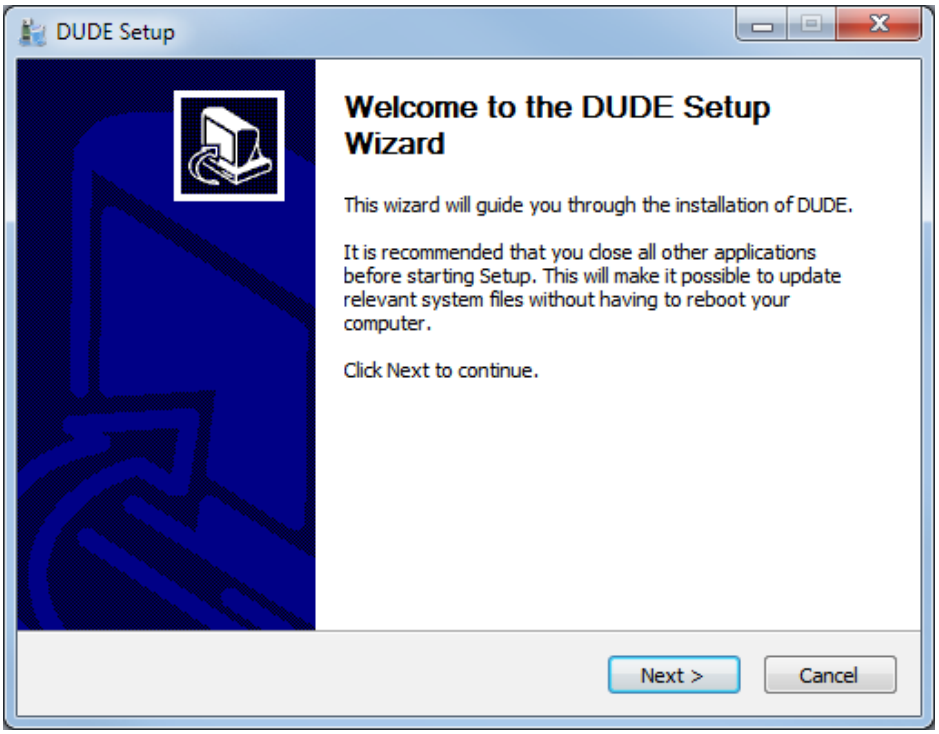
The "DUDE" driver supports two languages – English and Romanian. The client programs can choose/set the language of the COM server programmatically for the preferred language and after that the messages for errors or device statuses will return on this language. You can see the example of the usage of this feature in the demo programs.

Installation

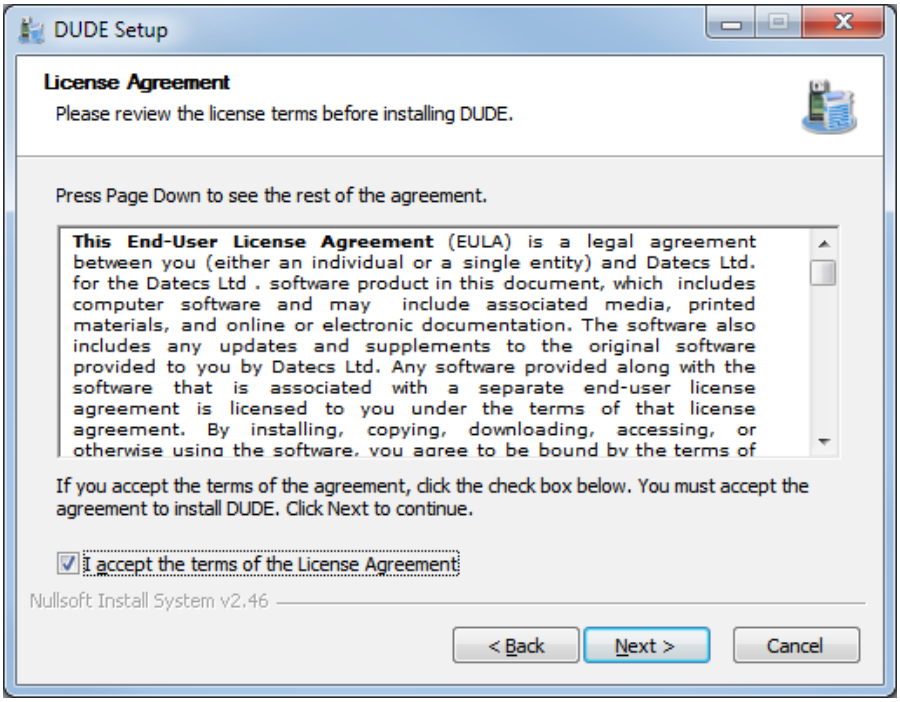
The installation of "DUDE" is as simple as possible. Below is given the installation process in English.

Double click on the setup file. If you see the dialog "User account control" which looks different depending on the Windows version – press the "Yes" button.

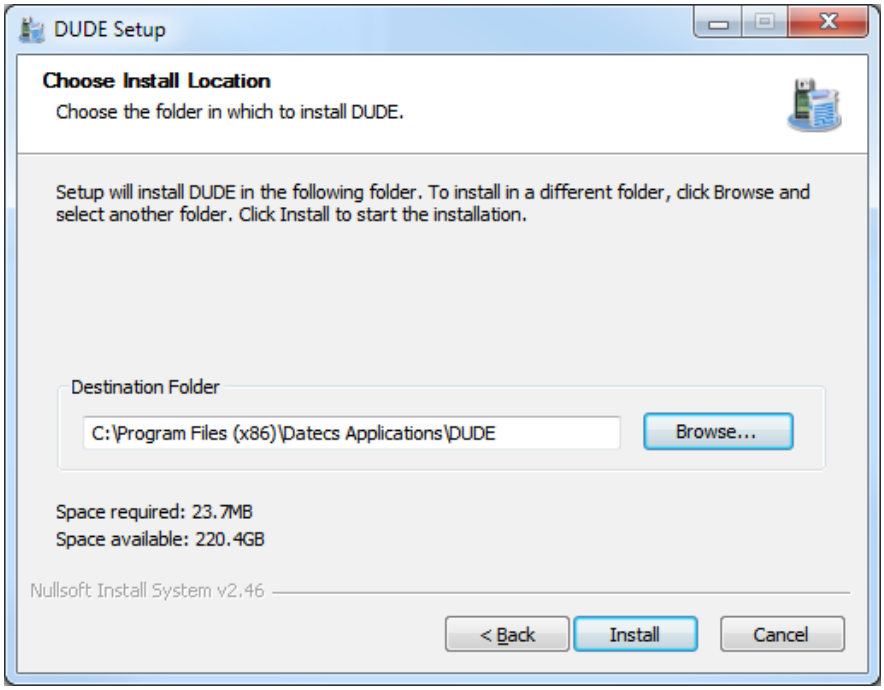
After the showing of the DUDE installer splash screen, you will see the installer language dialog. Choose the preferred language and click on the "Ok" button.



In the "Welcome screen" – click on the "Next" button. After that you will see the page with a "License Agreement".

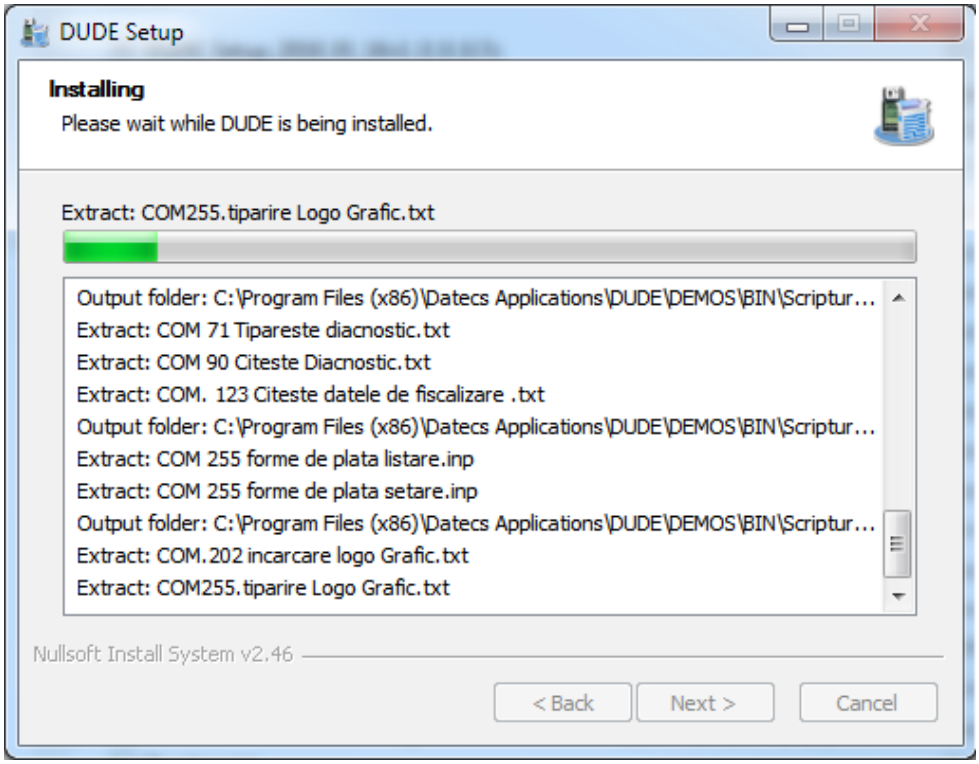


Read the license agreement carefully and if you agree with the terms of the agreement, click on the check-box and press the "Next" button.

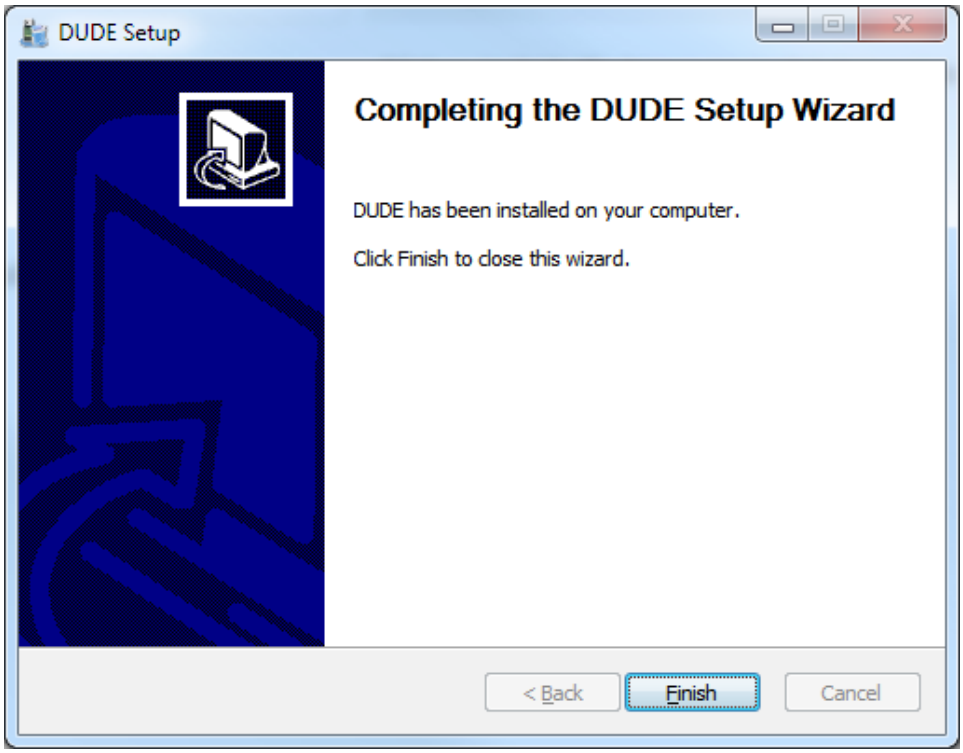


In the next page – you can change the destination folder. If you want to change the destination – type the proper destination into the edit box or click on the "Browse" button. In this case you will see the "Browse For Folder"dialog and can choose the preferred folder.

After that – click on the "Install" button.



Please wait for the installation to finish.



If everything is fine – you will see the final screen of the installer. Click on the "Finish" button.

That is all – the DUDE driver is installed on the PC. You dont need to start or set up anything. The needed settings of the COM server will be set programmatically by the programs that use the driver.

A few words about the low level communication protocol

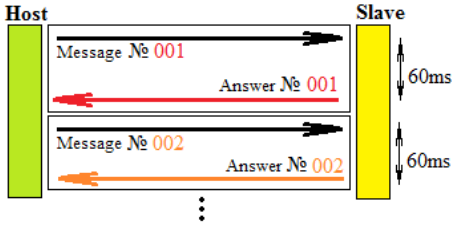
The DUDE COM server encapsulates the implementation of the described below low level communication protocol with the fiscal devices made by Datecs. You don‘t have to know the protocol in details but here we will provide some information about it. Sometimes, for the needs of development, you can choose to use the communication tracking mode of the server. If you want to send us a bug report for this level - it is helpful if you have some knowledge of the protocol.

Synchronous protocol

The protocol for communicating with Datecs fiscal devices is synchronous.

- The order for initiating, sending and receiving messages is synchronized;
- The time is synchronized;
- The size and format of messages are synchronized;

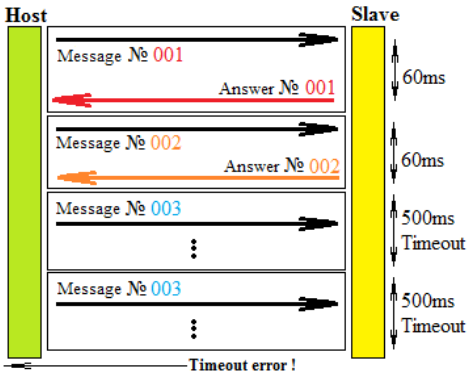
Synchronization of the order for initiating, sending and receiving messages



The fiscal device (Slave) cannot initiate communication. It is instead always initiated by the program located on the relevant computer (Host). Receiving information from a fiscal device through a communication channel without initiating it is typically a sign of a defective device or of an uncleaned buffer.

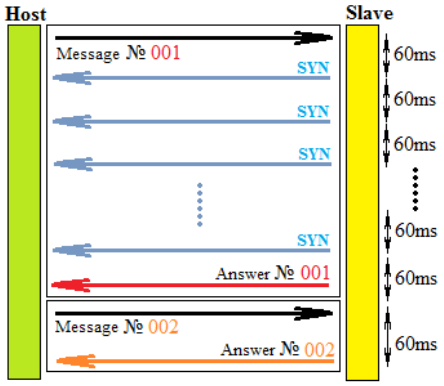
- Host initiates communication by sending a packaged message to Slave, containing a command;
- Depending on its current status, Slave may or may not execute the desired operation, after which it must reply with a packaged message (or with a single byte, in some cases);
- Host should wait for Slave’s response before sending another message, unless the timeout window has expired (see time-based synchronization);

Time-based Synchronization



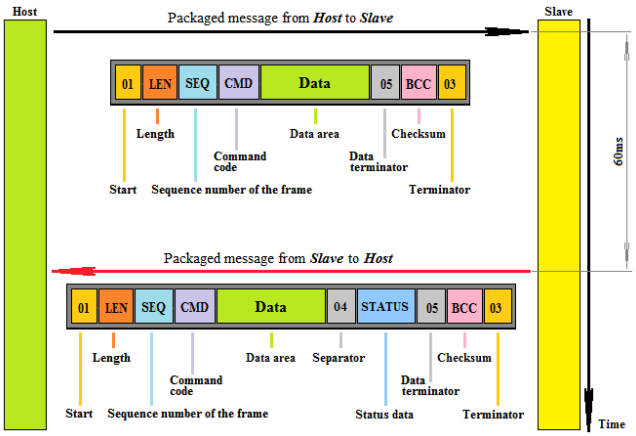
- To every command coming from the Host, Slave must respond no later than 60ms (with either a packaged message or with a specifically determined single-byte code);
- Host has a 500ms timeout for receiving Slave's reply;
- If Host does not receive a reply within 500ms, it should transmit the same message again, using the same sequence number and the same command;
- After two unsuccessful attempts, Host should indicate that there is either no connection with Slave or that there is a hardware malfunction within the device;
- A 500ms timeout window is used when communicating using RS-232 or USB. When using a different transport protocol (such as TCP/IP), a greater timeout window should be allowed, depending on the quality of the connection.

Support for a normal delay in a response from Slave



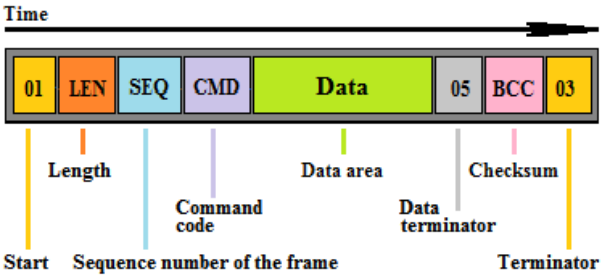
If the command is such that Slave will experience a delay when replying, Slave begins sending SYN bytes through the channel every 60ms. This way, Slave tells Host it is busy at present but is fully operational and in working order, i.e it is telling Host to wait a little.

Message format-based Synchronization



- All commands and replies between Host and Slave (aside from the single-byte messages described above) are packaged in a specifically determined format, known in advance; "Randomness" is allowed solely in the area of logical data but the location of this data in the package, as well as its maximum size, is strictly determined;
- The formats of the packaged message from Slave to Host and vice versa are specifically defined yet "different", as Slave's reply contains additional information on the status of the device. Even if the device has nothing to send Host in the form of logical data (i.e the data area is blank), it will still reply so that the program (Host) can, based on the status bytes, check the condition of the device and check for any issues (for example, running out of paper);
- In case of a problem or noise in the communication channel, it is entirely possible for Slave to receive a command with an erroneous control sum (checksum). In this case Slave replies with a single NAK byte. That indicates that Host should repeat its message without any changes to it – or simply resend it.

Packaged message description

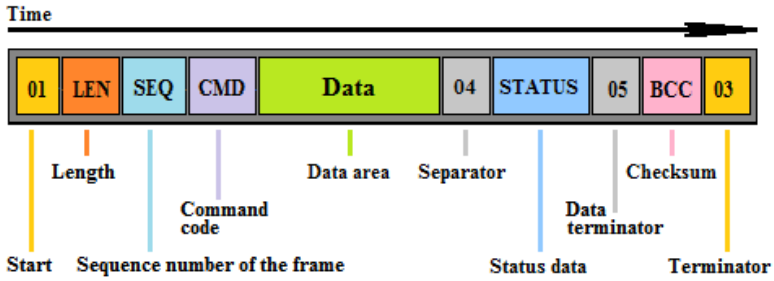


Packaged message from ***Host*** to ***Slave***

- **01** (**Start**) - Start byte.
  - Length: 1 byte;
  - Value: 01H;
- **LEN** – Length of message.

  This is the number of bytes from the start byte <01> (**excluding itself**) to the data terminator <05> (**including itself**), plus a fixed offset of 20H.
  - Length: 2 bytes (word), coded in 4 bytes;
  - Values: Since the possible values in this protocol are in the interval [0..65535] (DEC), your program should fill in the necessary value in the variable of type word. Every digit of these two bytes is transmitted and 30H is added to it. As a result, there will be four bytes in the packaged message for this field. The reason for this coding is related to the internal execution of the protocol in fiscal devices and is of no matter.

- Example: the variable with a value of 1AE3H in the packaged message is presented as four bytes with values as follows: 31H,3AH,3EH,33H;



Packaged message from *Slave* to *Host*

- **SEQ** – The sequence number of the packaged message. Slave records the same sequence number in its reply. If Slave receives a message with a value of identical to the one from the last message it has received, it does not perform an action but instead repeats the last message it has sent.
  - Length: 1 byte;
  - Values: from 20H to FFH;
- **CMD** – The number of the command.
  - If Slave receives non – existing or non – valid command code, it replies with a packaged message with a length of zero in the data field and raises the status bit for an invalid command to one;
  - When the Slave replying to a given message – the Slave use the same value of the command code in the packed message to the Host;
  - The upper boundary for the value of permissible commands depends on the given device;
  - Length: 2 bytes (word), coded in 4 bytes;
  - Values: Since the possible values in this protocol are in the interval [0..65535] (DEC), your program should fill in the necessary value in the variable of type word. Every digit of these two bytes is transmitted and 30H is added to it. As a result, there will be four bytes in the packaged message for this field. The reason for this coding is related to the internal execution of the protocol in fiscal devices and is of no matter;
  - Example: the variable with a value of 1AE3H in the packaged message is presented as four bytes with values as follows: 31H,3AH,3EH,33H;
  - Notice: *The list of specific, permitted and valid values of commands for a given fiscal device should be acquired from that specific device's documentation. That a given value is possible doesn't necessarily mean it is permitted or used;*
- **Data** – Logical data. The format and length of the data area depend on the command;
  - If the command doesn't require sending data to Slave, or if consequently Slave's response to Host has no data, then the length of this field is zero.
  - If when sending a command from Host to Slave there is a syntax error in the data, Slave replies with a packaged message with a zero-length data field and raises the value of status bit for syntax error to one;

- Length (Host to Slave): 0 – 213 bytes;
- Length (Slave to Host) 0 – 218 bytes;
- Values: 09H, 0AH and the values in the interval between 20H and FFH;

- **04** (**Separator**) - Separator. In the packaged message from Host to Slave, this is the byte which serves as a marker separating the logical data from the data on Slave's status (status bytes).
  - Length: 1 byte;
  - Value: 04H;
- **STATUS** – the data field containing information on the current status of the fiscal device. In subsequent articles I will refer to bytes from this field as "status bytes".
  - ***The meaning of every bit of every status byte should be acquired from the documentation of the relevant device***;
  - The condition of a given bit corresponds logically to a specific device condition. In subsequent articles I will call all such bits "status bit", and when referring to the condition of a given status we will refer to the value of the given status bit;
  - On the whole – bits can be categorized in the following groups and subgroups:
    - Ones which can be ignored:
      - A given status bit can be saved for future use (depending on the situation it can always have a value of 0 or 1);
      - A given status bit can be saved for internal use (depending on the situation it can always have a value of 0 or 1);
    - **Ones which should be paid special attention to**:
      - **Informative status bits** – they carry information on the logical or physical condition of the device. Some operations are not permitted in certain situations and therefore it would be prudent to monitor them well. For example:
        - If the printer is out of paper – it cannot print out any documents whatsoever;
        - If the printer isn't fiscalized – apparently it cannot issue fiscal receipts;
      - **Error status bits** – they carry general information on the error which has occurred. This can be a non-permitted command, a syntactic error etc. Sometimes (with the old protocol) the only way for the program to make a correct decision for resolving the situation is to assemble the information from all available statuses in the reply of Slave;
  - Length: 8 bytes;
  - Values for each of the bytes: from 80H to FFH;
- **05** (**Data Terminator**) - Separator.
  - In the packaged message from Host to Slave this is the byte which serves as a market, separating the logical data from the rest of the packaged message.
  - In the packaged message from Slave to Host this is the byte which serves as a market, separating the status data from the rest of the packaged message.
  - Value: 05H;

- Length: 1 byte;

- **BCC** - Checksum. The checksum includes the values from <01> (excluding itself) to <05> (including itself). Since the possible values in this protocol are in the interval [0..65535] (DEC), your program should fill in the necessary value in the variable of type word. Every digit of these two bytes is transmitted and 30H is added to it. As a result, there will be four bytes in the packaged message for this field. The reason for this coding is related to the internal execution of the protocol in fiscal devices and is of no matter;
  - Example: the variable with a value of 1AE3H in the packaged message is presented as four bytes with values as follows: 31H,3AH,3EH,33H;
  - Length: 2 bytes (word), coded in 4 bytes;
  - Values: from 30H to 3FH;
- **03** (**Terminator**) - A byte marking the end of the packaged message.
  - Length: 1 byte.
  - Value: 03H.

Using of the DUDE

A few words about the dependencies from the hardware

Device-dependent software refers to programs that can run only on a certain type of hardware. Device-independent software work right no matter what model of device you use them with.

DUDE has mixed capabilities in this area – for example the properties and the methods are the same for all models listed above but at the same time – some of the maximal or minimal values depend on the current device. For example: the maximum values of the printed symbols.

So – when you design your project you can use the DUDE as an universal and common solution for the fiscal devices of Datecs but you must be aware that in fact it is a device dependent SDK.

Thread safe or not?

One of the first questions about this COM server was "The COM Server is "Thread-safe" or not?".

The "Thread-safe" term sometimes is a very vague term. To answer the question, clearly we need to know what "Thread-safe" means: "A piece of code is thread-safe if it functions correctly during simultaneous execution by multiple threads."

The usage of this COM server is not "Thread-safe"! We do not manage the commands to the fiscal devices into a queue and if you send the commands to the fiscal device from multiple threads simultaneously – most probably some of the commands won‘t be executed correctly. The fiscal device itself will reject some of them. *For example: If you want to sale simultaniously from multiple threads but without managing your own queue – most probably the fiscal device will reject the second opening of the fiscal receipt. The reason is the restriction by law. You have no right to open a second fiscal receipt if the first one is not closed properly.*

The COM server works properly and sends commands to the fiscal device correctly – but you must to know about this peculiarity related to the usage of the server from multiple threads simultaneously and to **design your project according that info**.

When to start or stop the COM server and when to open or close the connection to the device?

The answer to these questions depends on the specifics of the programming language you are using and on the entire project design of your application.

Our own preferences and recommendations are (if this is possible for your project):

- Start the COM server upon the initialization of the program;
- Keep the connection open during the entire possible usage of the fiscal device. If this is possible – use the COM server and the connection to the device for only one thread (application).

Generally- there no problem in also:

- Starting the COM server at the beginning;
- If needed, opening and closing the connection to the device as many times as needed;
- Closing the COM server at the end of the usage of the program;

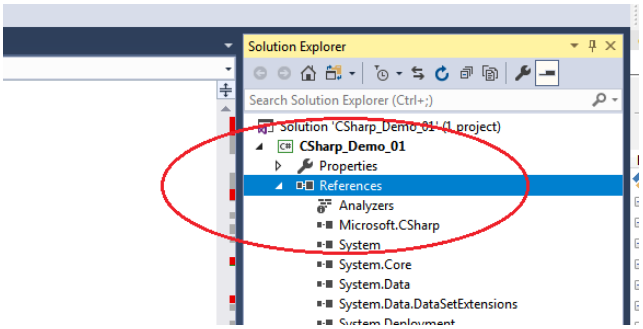
If you must use one fiscal device from many work places – our recommendation is to first create a tool (for example TCP/IP server) which will use the COM server by the described above algorithm. After that the other modules of your product will communicate with a fiscal device through this tool. This is also the place to manage properly the incoming orders (commands) which come simultaneously. Put them into a queue and so on.

If you are considering starting or stopping the COM server on every use of the fiscal device – please keep in mind the specifics of the programming language you use. In this case you must make sure that you always close the connection to the device and that the COM server is stopped completely before starting to use another instance of the COM server.

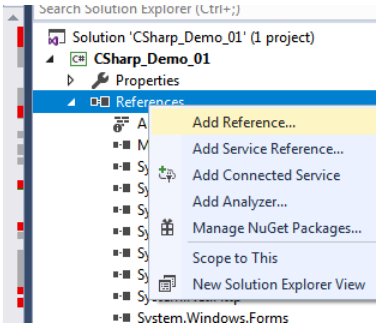
Using DUDE from Visual Studio 2017 (C#)

It is easy.

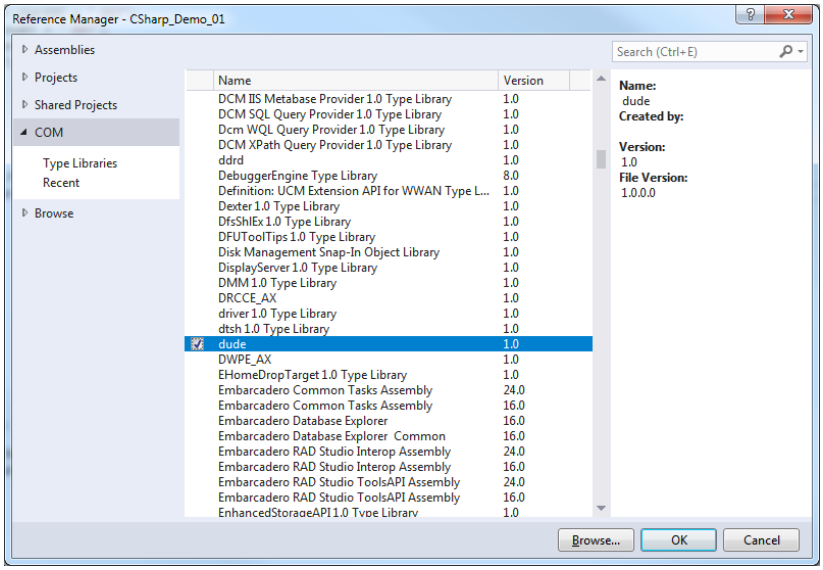
- ✔ Install DUDE and after that – go to the "Solution Explorer\References" in your project;



- ✔ Click the right mouse button and choose “Add reference” ;



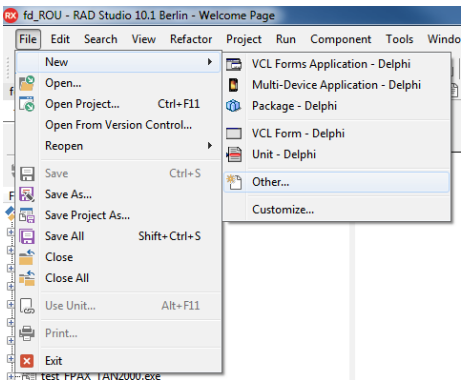
- ✔ In the dialog wizard – choose COM, find dude into the list and click “OK” button;



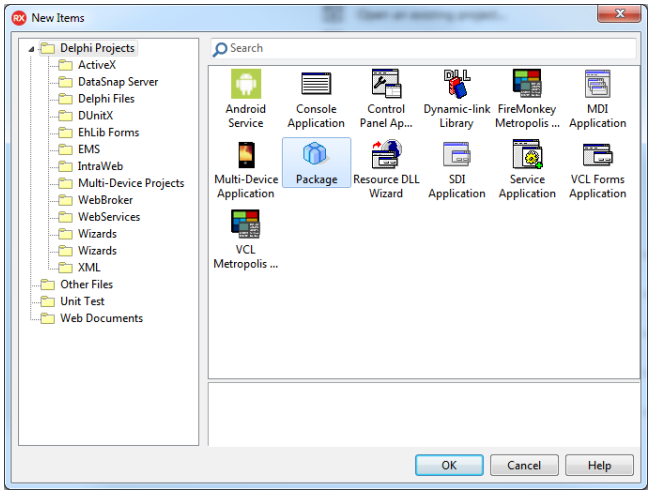
- ✔ That’s all – the studio will add the reference to the dude and you can start to use it;
- ✔ You can found an example demo program with source code into the installation folders;

Using DUDE from Delphi

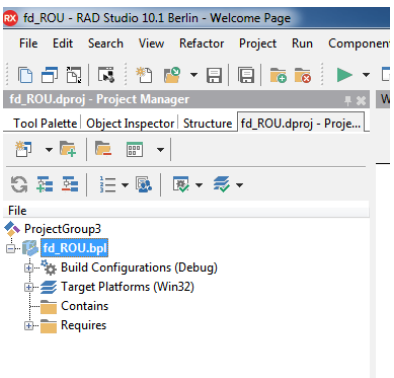
It is easy. There is more than one way to use a COM server under Delphi so I'll show you here the preferred by me method. I prefer to add the type library into a package and after registration of this package – we can use the DUDE as a component. The example is made under Delphi 10.1 Berlin but the way is the same for all other versions of Delphi – back to the Delphi 7.



✔ Go to “File\New” and choose “Other”;

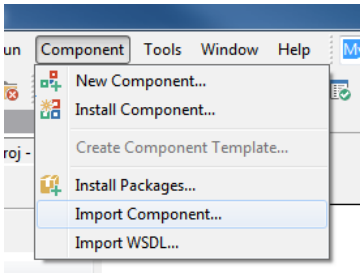


✔ Choose to create a new package and click “OK” button;

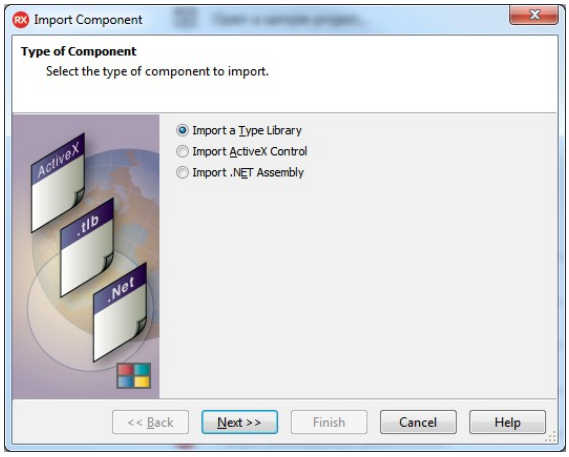


✔ Delphi will create a new blank package project so now you can save it as you prefer but for

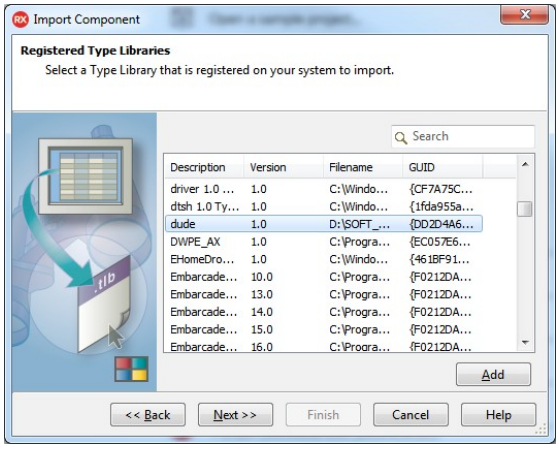
the needs of example – lets save the project as “fd_ROU.dproj” (from fiscal devices – Romania);



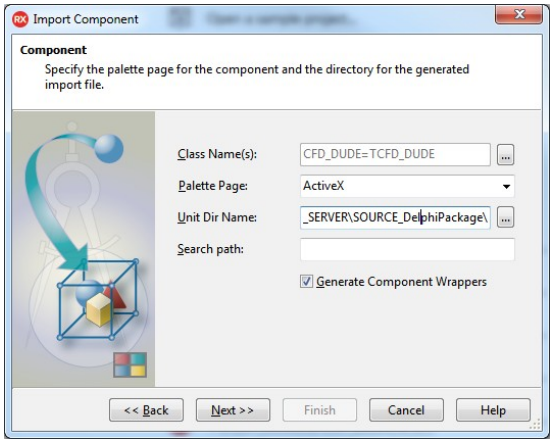
✔ Go to the menu and choose “Component\Import Component”;



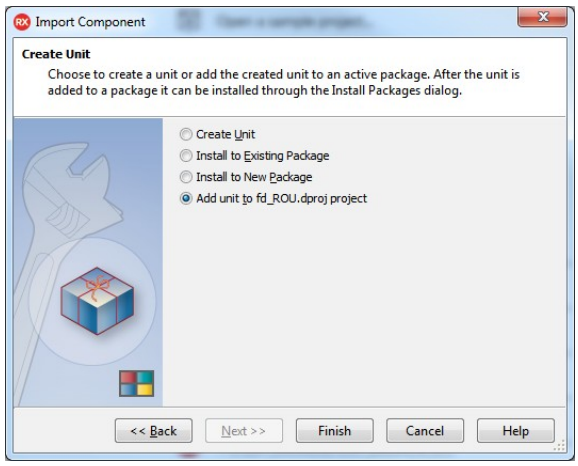
✔ Choose “Import a Type Library” and click “Next” button;



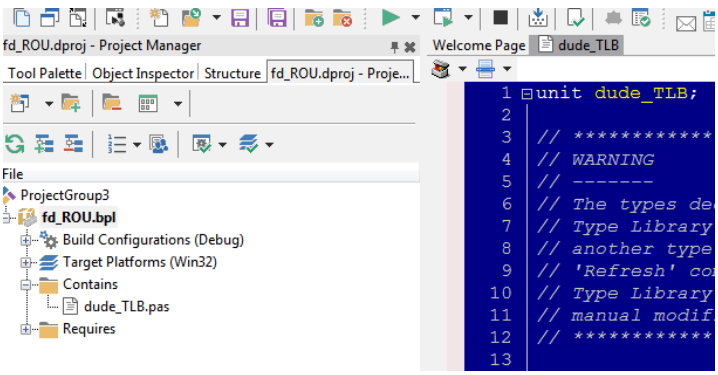
✔ On the next page – find dude into the list and click “Next” button;



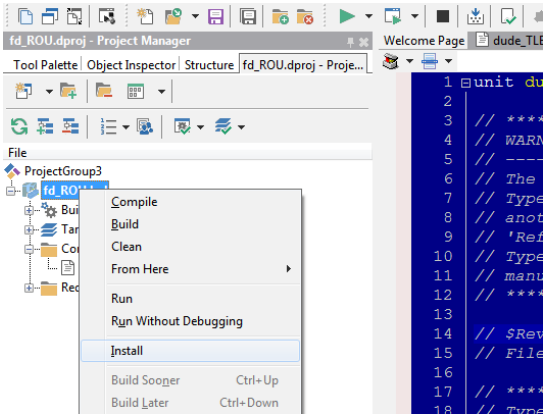
✔ On the next page – choose the “Palette page”, “Unit dir name”, activate the generation of the component wrappers and click “Next” button;



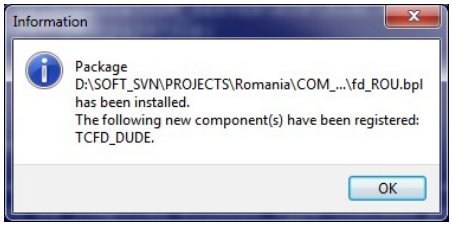
✔ Choose option “Add unit to fd_ROU.dproj project” and click “Finish” button;

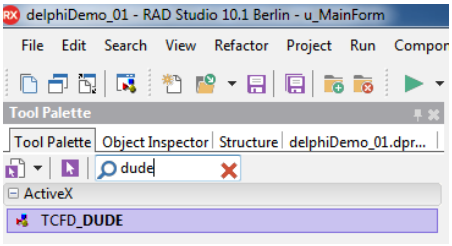


✔ Delphi will create “dude_TLB.pas” for you. Save and Build the project;

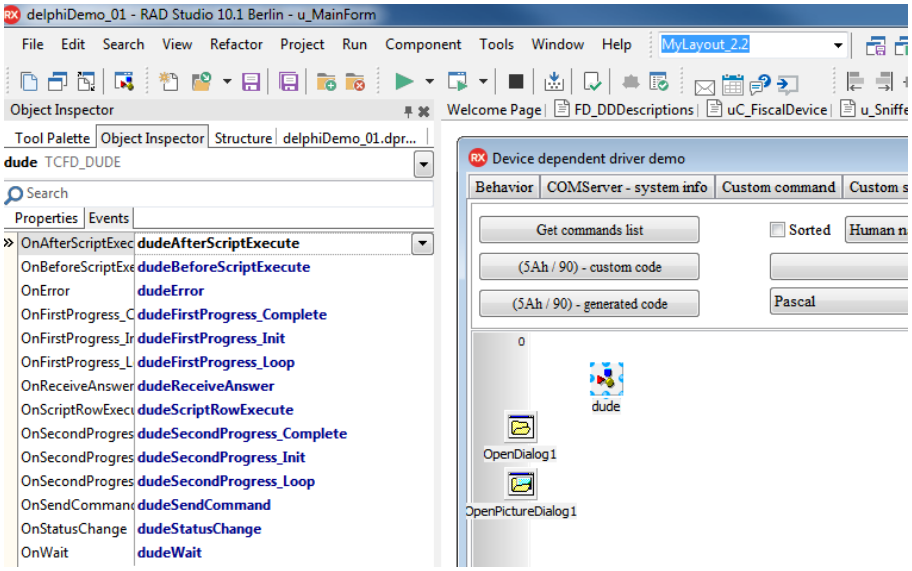


✔ Install the package into the IDE and you can start to use the DUDE from other projects;





✔ You can found you new component into the “Tool Palette” and start using it;



✔ You can found an example demo program with source code into the installation folders;

✔ That’s all;

Note: The demo project use JVCL/JCL source code that can be obtained from: https://github.com/project-jedi,

https://github.com/TurboPack/SynEdit*, or from the "GetIt Package manager" into the newest versions of Delphi.*

Enumerated types

DUDE – COM Server exports some enumerated types which are used internally and also as a more human oriented way to set some parameters by client applications.

Enumerated type name	Values	Description
TDudeLanguage	• English • Romanian	The client programs can choose/set the language of the COM server programmatically for the preferred language and after that the messages for errors or device statuses will return in this language.
TTransportProtocol	• ctc_RS232 • ctc_TCPIP	The client application must set the type of the communication transport protocol before opening the connection to the fiscal device.
TDeviceType	• dt_FiscalPrinter • dt_ECR	Used only for informative purposes. After a successful connection to the fiscal device the DUDE driver can tell you that this device is fiscal printer or cash register.
TDeviceModel	• mc_Unknown • mc_DP_05 • mc_DP_25 • mc_DP_35 • mc_DP_150 • mc_WP_50 • mc_WP_500 • mc_FP_650 • mc_FP_700 • mc_FP_800 • mc_FMP_350 • mc_FP_650_V • mc_FP_700_V • mc_FP_800_V • mc_FP_2000	After a successful connection to the fiscal device the client application can read from DUDE the model of the fiscal device. This type of information is used in those cases when some values are device dependent. For example - the maximum of the printed columns.
TScriptType	• DS • FPrint	DUDE has a method through which hum the client application can execute pre-prepared text scripts. This type is used when the script engine must to know the language of the incoming script. ***Note**: In this version of DUDE - the FPrint script type is not supported.*
TCodeType	• Delphi • CSharp	DUDE can execute commands from client applications to the fiscal device by three ways: • By using the method "execute_Command" • By using the method "execute_Script_V1" • By using the method "execute_Command_ByName" DUDE driver can generate an example source code on two languages for all commands which are exported from the method "get_ComandsList". This source code is an example of the usage of the method "**execute_Command_ByName**" and you can use it if you find it proper or usable. ***Note**: In this version of DUDE – the driver generate source code only on Pascal (Delphi) and C#.*

Properties

Device status – error status bits

The current status of the device is coded in an 8-byte field which is returned within each one packaged message from the fiscal device. The properties listed below return the condition of these flags which the fiscal devices fired in case of error or malfunction. You can read and must use the status of these properties:

- After a connection with the device is successfully established;
- After execution of a command or script;

Property name	Property type	Status [byte,bit]	Human meaning
eSBit_GeneralError_Sharp	Boolean Read only	[0,5]	General error - this is OR of all errors marked with #
eSBit_PrintingMechanism	Boolean Read only	[0,4]	# Failure in printing mechanism.
eSBit_ClockIsNotSynchronized	Boolean Read only	[0,2]	The real time clock is not synchronized.
eSBit_CommandCodeIsInvalid	Boolean Read only	[0,1]	# Command code is invalid.
eSBit_SyntaxError	Boolean Read only	[0,0]	# Syntax error.
eSBit_CommandNotPermitted	Boolean Read only	[1,1]	# Command is not permitted.
eSBit_Overflow	Boolean Read only	[1,0]	# Overflow during command execution.
eSBit_EJIsFull	Boolean Read only	[2,2]	EJ is full.
eSBit_EndOfPaper	Boolean Read only	[2,0]	# End of paper.
eSBit_FM_NotFound	Boolean Read only	[4,6]	Fiscal memory not found or damaged.
eSBit_FM_NotAccess	Boolean Read only	[4,0]	* Error when trying to access data stored in the FM.
eSBit_FM_Full	Boolean Read only	[4,4]	* Fiscal memory is full.
eSBit_GeneralError_Star	Boolean Read only	[4,5]	OR of all errors marked with ‘*’

Device status – informative status bits

The current status of the device is coded in an 8-byte field which is returned within each one packaged message from the fiscal device. The properties listed below return the condition of some of these flags which have only an informative meaning and can not be considered as a message for error or malfunction.

You can read and must use the status of these properties:

- After successfully establishing connection with the device;
- After execution of a command or script;

Property name	Property type	Status [byte,bit]	Human meaning
iSBit_Cover_IsOpen	Boolean Read only	[0,6]	Cover is open.
iSBit_No_ClientDisplay	Boolean Read only	[0,3]	No client display connected.
iSBit_24h_AfterDayOpening	Boolean Read only	[1,2]	More than 24 hours after day opening.
iSBit_Receipt_Nonfiscal	Boolean Read only	[2,5]	Nonfiscal receipt is open.
iSBit_EJ_NearlyFull	Boolean Read only	[2,4]	EJ nearly full.
iSBit_Receipt_Fiscal	Boolean Read only	[2,3]	Fiscal receipt is open.
iSBit_Near_PaperEnd	Boolean Read only	[2,1]	Near paper end.
iSBit_LessThan_60_Reports	Boolean Read only	[4,3]	There is space for less than 60 reports in Fiscal memory.
iSBit_Number_SFM_Set	Boolean Read only	[4,2]	Serial number and number of FM are set.
iSBit_Number_Tax_Set	Boolean Read only	[4,1]	Tax number is set.
iSBit_VAT_Set	Boolean Read only	[5,4]	VAT are set at least once.
iSBit_Device_Fiscalized	Boolean Read only	[5,3]	Device is fiscalized.
iSBit_FM_formatted	Boolean Read only	[5,1]	FM is formatted.

Information about connected device

You can read and use these properties:

- After successfully establishing connection with the device;
- After execution of a command or script;

Property name	Property type	Human meaning
connected_ToDevice	Boolean Read only	True if the connection to the fiscal device is successfully established.
support_RS232	Boolean Read only	True if the device supports communication through RS232 (or USB).
support_TCPIP	Boolean Read only	True if the device supports communication through LAN connector (TCP/IP protocol).
device_Type	TDeviceType Read only	• dt_FiscalPrinter for a fiscal printer; • dt_ECR for a cash register;
device_Number_Serial	WideString/BSTR Read only	Manufacturer number (serial number) of the device.
device_Number_FMemory	WideString/BSTR Read only	Fiscal number of the device.
device_Distributor	WideString/BSTR Read only	Distributor company name.
device_Model	TDeviceModel Read only	• mc_DP_05 for model DP-05 • mc_DP_25 for model DP-25 • mc_DP_35 for model DP-35 • mc_DP_150 for model DP-150 • mc_WP_50 for model WP-50 • mc_WP_500 for model WP-500 • mc_FP_650 for model FP-60 • mc_FP_700 for model FP-700 • mc_FP_800 for model FP-800 • mc_FMP_350 for model FMP-350 • mc_FP_650_V • mc_FP_700_V • mc_FP_800_V • mc_FP_2000 • mc_Unknown if the server can not recognise the connected device
device_Model_Name	WideString/BSTR Read only	The model name received from the device.
codePage	Integer/Long Read only	The code page used from the device.
device_Firmware_Revision	WideString/BSTR Read only	Device firmware revision.
device_Firmware_Date	WideString/BSTR Read only	Device firmware date and time.
device_Firmware_CheckSum	WideString/BSTR Read only	Device firmware checksum.

Communication

This section contains information about the properties which are related to the behavior of the COM server when you try to open a connection or during communication with the fiscal device.

Property name	Property type	Human meaning
protocol_TransportType	TTransportProtocol Read only	The client application must set the type of the transport protocol with execution of method "set_TransportType" before opening of the connection to the fiscal device. • ctc_RS232 – if connection is via (RS-232/USB) • ctc_TCPIP – if connection is via LAN (TCP/IP)
tcpip_Address	WideString/BSTR Read only	Current IP address of the fiscal device.
tcpip_Port	Integer/Long Read only	Current value of the TCP/IP port from the device side.
rs232_ComPort	Integer/Long Read only	Current COM port value.
rs232_BaudRate	Integer/Long Read only	Current baud rate.
read_TimeOutValue	Word Read/Write	Global timeout value for communication with the device. Default values: • 1000 mSec for RS-232 • 3000 mSec for TCP/IP
exit_ByReadTimeOutIsOn	Boolean Read/Write	If this value is true and if there is no answer for period of time bigger than global timeout value in mSec - DUDE will stop waiting for a response from the fiscal device and will return a timeout error code for current command execution. Default value is true.
rs232_ReadIntervalTimeout	Integer/Long Read only	Part of commtimeouts structure. The maximum time allowed to elapse before the arrival of the next byte on the communications line, in milliseconds. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used. A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the bytes that have already been received, even if no bytes have been received.
rs232_ReadTotalTimeoutMultiplier	Integer/Long Read only	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.

rs232_ReadTotalTimeoutConstant	Integer/Long Read only	Part of commtimeouts structure. A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes. A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.
rs232_WriteTotalTimeoutMultiplier	Integer/Long Read only	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.
rs232_WriteTotalTimeoutConstant	Integer/Long Read only	Part of commtimeouts structure. A constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written. A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.
rs232_OnOpen_Set_DCB	Boolean Read/Write	When this value is true – DUDE will try to set DCB structure during the opening of the COM port. Default value is true.
rs232_OnOpen_Set_DTR_ToFalse	Boolean Read/Write	When this value is true – DUDE will try to set DTR to false during the opening of the COM port. Clears the DTR (data-terminal-ready) signal. Default value is true.
rs232_OnOpen_Set_RTS_ToFalse	Boolean Read/Write	When this value is true – DUDE will try to set RTS to false during the opening of the COM port. Clears the RTS (request-to-send) signal. Default value is true.
connected_ToLAN	Boolean Read only	A return value of TRUE indicates that either the modem connection is active, or a LAN connection is active and a proxy is properly configured for the LAN. It does not guarantee that a connection to a specific host can be established. A return value of FALSE indicates that neither the modem nor the LAN are connected.

Behavior

Related to the commands execution

Property name	Property type	Human meaning
active_OnBeforeScriptExecute	Boolean Read only	If this value is true – DUDE will fire an event before the execution of the script.
active_OnScriptRowExecute	Boolean Read only	If this value is true – DUDE will fire an event after execution of the command from the text row into the script.
active_OnAfterScriptExecute	Boolean Read only	If this value is true – DUDE will fire an event after the execution of the script.
active_OnSendCommand	Boolean Read only	If this value is true – DUDE will fire an event after the sending of packet message to the fiscal device.
active_OnWait	Boolean Read only	If this value is true – DUDE will fire an event after receiving SYN byte from the fiscal device.
active_OnReceiveAnswer	Boolean Read only	If this value is true – DUDE will fire an event after receiving of packet message from the fiscal device.
active_OnStatusChange	Boolean Read only	If this value is true – DUDE will fire an event after receiving a packet message from the fiscal device.
active_OnError	Boolean Read only	If this value is true – DUDE will fire an event if an error occurs.

Related to the loops support

Property name	Property type	Human meaning
active_OnFirstProgress_Init	Boolean Read only	If this value is true – DUDE will fire an event before starting an operation which is a loop of commands. It is usable for initialization of progress bar for example.
active_OnFirstProgress_Loop	Boolean Read only	If this value is true – DUDE will fire an event after the execution of a command during to the loop of commands.
active_OnFirstProgress_Complete	Boolean Read only	If this value is true – DUDE will fire an event after finishing an operation which is a loop of commands.
active_OnSecondProgress_Init	Boolean Read only	If this value is true – DUDE will fire an event before starting an operation which is a loop of commands. It is usable for initialization of progress bar for example.
active_OnSecondProgress_Loop	Boolean Read only	If this value is true – DUDE will fire an event after the execution of a command during to the loop of commands.
active_OnSecondProgress_Complete	Boolean Read only	If this value is true – DUDE will fire an event after finishing an operation which is a loop of commands.

RAO mode

Changing of the "Register ActiveX Object on start" will change the behavior of the COM server ***after the restart of the COM server***.
If you use "Register Activex Object on start" with value "True":

- The instance and connection to the device will be "shared" between all connected client applications. For example when some of the applications execute a command – the events will be shared also. You will receive the events from the execution of the commands from other client applications. So in this mode the client applications must be designed for such behavior;
- Your client applications can check for an old running instance of the COM server and can choose to try to destroy the old instance of the COM object and after that to create/start a new instance of the COM object
- Your client applications can check for an old running instance of the COM server and can choose to connect and use the "shared" connection to the device;

When the client application was killed as a process from the task manager (or due to some crash) - the running instance don't know about this event. The COM port is still opened. If your application don't use "RAO mode" but the customer restart the application (not Windows only the app) - the new instance of the "dude" can't use this com port. But… if you use the "RAO mode" your client applications can check for an old running instance of the COM server and can choose to try to destroy the old instance of the COM object before to start using "Dude". Or of course to connect and use the connection to the device. *We can't recommend you which strategy is the right*. "Dude" is designed for a "normal" usage which means – without crashes of the client applications and so on but as programmers we know that it happens. So... you have to keep all this in mind and choose the right strategy for your own case. It really depends from the case. For example if your auto-update system just kill and update the application in the midnight but do not restart the Windows the "RAO" mode is for you.

- For how to use – check the source code under the button "Search and destroy" (Delphi demo) or write me a letter:
  - To check for an old instance – use GetActiveOleObject('dude.CFD_DUDE');
  - To destroy the old instance – use the method "DestroyInstance":
    - If you sure that only yours application use the COM server on the current machine;
    - Use this method only in the cases when the client application was killed from some reasons from the task manager (or some crash) and "Dude" don't know about that;
    - **Do not use the method "DestroyInstance" for a normal cases (directly)** – this method is designed only for the cases described above!

Property name	Property type	Human meaning
register_ActiveObject_OnStart	Boolean Read only	By default the value of this setting is false. If the value is true – DUDE will try to register ActiveX object into the Running Object Table (ROT).
save_SettingsAfterOpenConnection	Boolean Read only	If the value is true – DUDE will try to save the settings after the successful connection to the device.
active_OnAfterOpenConnection	Boolean Read only	If this value is true – DUDE will fire an event after successful connection to the device.
active_OnAfterCloseConnection	Boolean Read only	If this value is true – DUDE will try to fire an event after closing the connection to the device.
active_OnAfterSettingsChange	Boolean Read only	If this value is true – DUDE will fire an event after changing of the settings. *Recommendation:* *Use it if "register_ActiveObject_OnStart" is with value "True".*

Other

Property name	Property type	Human meaning
language	TDudeLanguage Read only	• English • Romanian
trackingMode	Boolean Read only	If the value is true – DUDE will try to save the communication with the fiscal device in a file according to the values of other tracking properties.
trackingMode_Path	WideString/BSTR Read only	The path to the log file when DUDE is in tracking mode.
trackingMode_FileName	WideString/BSTR Read only	If DUDE is in tracking mode – it will try to create and save the log info a text file with this name.
trackingMode_RowLimit	Integer/Long Read only	This value is a limit of the number of text rows into the log file. DUDE accept values between 100 and 5000.
lastError_Code	Integer/Long Read only	Contains the current value of the last error code of the DUDE.
lastError_Message	WideString/BSTR Read only	Contains the current value of the last error message of the DUDE.
last_AnswerList	WideString/BSTR Read only	Contains the current value of the last answer from the packaged message received from the fiscal device. The values are separated with CRLF. (bytes with the decimal values 13 and 10. "Carriage Return" and "Line Feed").
download_Path	WideString/BSTR Read only	Contains the current value of the path for download which the DUDE driver will try to use in operations for downloading ANAF files from the fiscal device.
zRange_StartValue	Integer/Long Read/Write	The method for downloading of ANAF files by Z-reports range use this value as a start value for the range. The client application must set this value properly before execution of method "download_ANAF_ZRange".
zRange_EndValue	Integer/Long Read/Write	The method for downloading of ANAF files by Z-reports range use this value as an end value for the range. The client application must set this value properly before execution of method "download_ANAF_ZRange".
DateRange_StartValue	WideString/BSTR Read/Write	The method for downloading of ANAF files by date range use this value as a start value for the range. The client application must set this value properly before execution of method "download_ANAF_DTRange". Date and time format: DD-MM-YY hh:mm:ss DST • DD – Day • MM – Month • YY – Year • hh – Hours • mm – Minutes • ss – Seconds • DST - Text DST. If present means that daylight saving time is active.
DateRange_EndValue	WideString/BSTR Read/Write	The method for downloading of ANAF files by date range use this value as an end value for the range.

		The client application must set this value properly before execution of method "download_ANAF_DTRange". Date and time format: DD-MM-YY hh:mm:ss DST • DD – Day • MM – Month • YY – Year • hh – Hours • mm – Minutes • ss – Seconds DST - Text DST. If exist means that summer time is active.

Events

By design, the events are intended to be used for informative purposes or for management of the user interface. During the execution of commands more than one event can be raised. Do not use the error codes from events related to handling of the commands execution!

OnError event

DUDE will fire this event when an error occurs.

Note: If the execution of the command is unsuccessful – the value of "**error_Code**" most probably will be equal to the result of the execution of given method but ***if the command is successfully executed – the result and last error code will be with a value of zero and this event won‘t be raised.***

Parameters:

Name	Type	Description
error_Code	Integer	The value of the error code depending of the error.
error_Message	WideString	The text error message.

OnBeforeScriptExecute event

DUDE will fire this event just before the execution of the script.
Parameters: **None**

OnScriptRowExecute event

DUDE will fire this event after the execution of the command from the text row into the script.
Parameters:

Name	Type	Description
row_Index	Integer	The index shows the line number depending on the beginning of the script. The index is zero based.
error_Code	Integer	The value of the error code depending of the error. An error code with value 0 means that there no error during the execution of the command.
input_Value	WideString	The value of the executed text line.
output_Value	WideString	The answer from the device side.

OnAfterScriptExecute event

DUDE will fire this event after the execution of the script.

Parameters: **None**

OnSendCommand event

DUDE will fire this event after sending of packet message to the fiscal device.

Parameters:

Name	Type	Description
Command	WideString	The value of the command in decimal.
DateAndTime	WideString	Date and time of execution in format 'dd.mm.yyyy hh:mm:ss:zzz'
repeat_Value	WideString	By the low level communication protocol – if command failed or if fiscal device asked the host to repeat the command, the host must send the same packet message to the fiscal device. The value of this parameter contains the current index of the attempts.
hex_Header	WideString	The header part from the packet message to the fiscal device in hex values.
hex_Data	WideString	The logical data part from the packet message to the fiscal device in hex values.
hex_Footer	WideString	The footer part from the packet message to the fiscal device in hex values.
human_Data	WideString	The logical data part from the packet message to the fiscal device before to convert into a hex values.

OnWait event

DUDE will fire this event after receiving a SYN byte from the fiscal device.

Parameters:

Name	Type	Description
Value	Byte	Must contain value equal to SYN

OnReceiveAnswer event

DUDE will fire this event after the receiving of packet message from the fiscal device.

Parameters:

Name	Type	Description
Command	WideString	The value of the command in decimal.
DateAndTime	WideString	Date and time of execution in format 'dd.mm.yyyy hh:mm:ss:zzz'
repeat_Value	WideString	By the low level communication protocol – if command failed or if fiscal device asked the host to repeat the command, the host must send the same packet message to the fiscal device. The value of this parameter contains the current index of the tries.
hex_Header	WideString	The header part from the packet message from the fiscal device in hex values.
hex_Data	WideString	The logical data part from the packet message from the fiscal device in hex values.
hex_Footer	WideString	The footer part from the packet message from the fiscal device in hex values.
human_Data	WideString	The logical data part from the packet message from the fiscal device as a human text.

OnStatusChange event

DUDE will fire this event after the receiving the packet message from the fiscal device. This event is usable as a trigger. After the raising of the event - the client application can get the status of the fiscal device from the status bytes (or from the status properties) because all of them have already been refreshed internally.

Parameters: **None**

OnFirstProgress_Init event

DUDE will fire this event before starting an operation which is a loop of commands. It is usable for initialization of progress bar, for example.

Parameters:

Name	Type	Description
value_Minimum	Integer	The calculated minimum value for the initialization.
value_Maximum	Integer	The calculated maximum value for the initialization.
value_Position	Integer	Current position for the initialization.

OnFirstProgress_Loop event

DUDE will fire this event after the execution of a command during the loop of commands.

Parameters:

Name	Type	Description
value_Position	Integer	Current position of the progress.

OnFirstProgress_Complete event

DUDE will fire this event after finishing an operation which is a loop of commands.

Parameters: **None**

OnSecondProgress_Init event

DUDE will fire this event before starting an operation which is a loop of commands. It is usable for initialization of progress bar, for example.

Parameters:

Name	Type	Description
value_Minimum	Integer	The calculated minimum value for the initialization.
value_Maximum	Integer	The calculated maximum value for the initialization.
value_Position	Integer	Current position for the initialization.

OnSecondProgress_Loop event

DUDE will fire this event after the execution of a command during the loop of commands.

Parameters:

Name	Type	Description
value_Position	Integer	Current position of the progress.

OnSecondProgress_Complete event

DUDE will fire this event after finishing an operation which is a loop of commands.

Parameters: **None**

Methods

Tracking mode

For the needs of development DUDE supports an internal engine for tracking communication. If the programmers want to know or use the communication between the host (PC) and the slave (the fiscal device) – they can activate this mode and DUDE will save the communication in the given file. Sometimes if you think you have found a bug or a problem – the support team will probably ask you to activate the tracking mode, to perform the same actions step by step and then to send the log file for analysis.

set_TrackingMode_RowLimit

Execution of this function will set the limit of the number of text rows into the log file. DUDE accept values between 100 and 5000.

Parameters:

Name	Type	Description
Value	Integer Long	The limit of the number of text rows into the log file. DUDE accept values between 100 and 5000.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_TrackingMode_Path

Execution of this function will set the path to the log file when DUDE is in tracking mode.

Parameters:

Name	Type	Description
Value	WideString BSTR	The path to the log file when DUDE is in tracking mode
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_TrackingMode_FileName

Execution of this function will set the log file name which DUDE will try to create and fill when it is in tracking mode.

Parameters:

Name	Type	Description
Value	WideString BSTR	The log file name
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_TrackingMode

Execution of this function will activate or deactivate the tracking mode. If the value is true – DUDE will try to save the communication with the fiscal device into a file according to the values of other tracking properties.

Parameters:

Name	Type	Description
Value	Boolean	If the value is true, the tracking mode will be activated.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

Communications

set_TransportType

Execution of this function will set the transport protocol type. The client application must set the type of the communication transport protocol before opening of the connection to the fiscal device.

Parameters:

Name	Type	Description
Value	TTransportProtocol	The client application must set the type of the transport protocol before opening the connection to the fiscal device. • ctc_RS232 – if connection is via (RS-232/USB) • ctc_TCPIP – if connection is via LAN (TCP/IP) For current value – check the value of the corresponding property: "protocol_TransportType".
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_TCPIP

Execution of this function will set the values needed for successful execution of the method "open_Connection" if the chosen protocol type is ctc_TCPIP. The client application must set the type of the communication transport protocol, the IPAddress and the Port before trying to open the connection to the fiscal device.

Parameters:

Name	Type	Description
IPAddress	WideString BSTR	The IP Address of the fiscal device.
Port	Integer / Long	The Port number for TCP/IP communications of the fiscal device.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_RS232

Execution of this function will set the values needed for successful execution of the method "open_Connection" if the chosen protocol type is ctc_RS232. The client application must set the type of the communication transport protocol, the ComPort and the BaudRate before trying to open the connection to the fiscal device.

Parameters:

Name	Type	Description
ComPort	Integer / Long	The ComPort.
BaudRate	Integer / Long	The baud rate – must be the same as into the fiscal device.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_RS232_Timeouts

Execution of this function will set the values of the commtimeouts structure. Do not use it if you don't know how or if DUDE is working properly with the default values.

Parameters:

Name	Type	Description
ReadIntervalTimeout	LongWord	Part of commtimeouts structure. The maximum time allowed to elapse before the arrival of the next byte on the communications line, in milliseconds. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used. A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the bytes that have already been received, even if no bytes have been received.
ReadTotalTimeoutMultiplier	LongWord	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.
ReadTotalTimeoutConstant	LongWord	Part of commtimeouts structure. A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes. A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.
WriteTotalTimeoutMultiplier	LongWord	Part of commtimeouts structure. The multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.
WriteTotalTimeoutConstant	LongWord	Part of commtimeouts structure. A constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written. A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

rs232_COMPortList

This method returns a list of visible for DUDE engine COM ports. If your COM port is not here – most probably the DUDE engine isn‘t using it properly. After the adding of new USB device or hardware (new COM port) – this method must be executed again to refresh the info of the client application.

Parameters:

Name	Type	Description
Result	WideString BSTR	This method returns a list of visible for DUDE engine COM ports.

open_Connection

This function opens a connection to the fiscal device according to the other communication properties set before the execution of the command.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

close_Connection

This function closes the connection to the fiscal device.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

Error support

You can find a full list of the possible errors returned from the DUDE engine or from the fiscal device in the document: "ErrorCodes.xls". This document is located in the folder DOCUMENTATION, which is a sub-folder of the root of the installation folder.

get_ErrorMessageByCode

The client applications can get the error message by given code.

Parameters and result:

Name	Type	Description
Value	Integer/Long	The value of the code.
Result	WideString BSTR	The error message for a given code.

get_Sbit_State

The client applications can get the state of any given status bit after each answer of the fiscal device.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	Boolean	The result = true – mean that the status bit is raised. *Example:* *if get_SBit_State(0,0) then ShowMessage('Syntax error!');*

get_Sbit_Description

The client applications can get the description text for any given status bit.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	WideString BSTR	The description text for the given status bit. *Example: ShowMessage(get_SBit_Description(0,0));*

get_Sbit_ErrorChecking

The client applications can get the description text for any given status bit.

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Result	Boolean	If the result of the function is true - this means that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_Sbit_ErrorChecking

The client application can change behavior of the DUDE engine according to the needs of the project. If you think that the raising of some status bit is not an error – you can switch off its checking.

Note: Change of the checking mean that only the DUDE engine will stop checking given status. The fiscal device will continue raising the given status bit and will continue to return errors less than zero if command is not apropriate or if there is a syntax error, for example. ***This function is developed for the needs of service applications and We do not recomend you to use it if you are not sure what we are talking about.*** *One more time: If the fiscal device "thinks" that given command is an error – it will not execute the command.*

Parameters and result:

Name	Type	Description
byteIndex	Byte	The index of the target status byte.
bitIndex	Byte	The index of the target status bit.
Value	Boolean	True or False if you want to switch on or switch off the checking of given status bit.
Result	Boolean	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

Behavior

The client applications can change the behavior of the DUDE in some aspects. For example they can activate or deactivate the raising of the events in the different cases.

set_ScriptEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnBeforeScriptExecute**;
- **OnScriptRowExecute**;
- **OnAfterScriptExecute**;

Parameters and result:

Name	Type	Description
active_OnBeforeScriptExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnScriptRowExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnAfterScriptExecute	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the DUDE engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_FirstProgressEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnFirstProgress_Init**;
- **OnFirstProgress_Loop**;
- **OnFirstProgress_Complete**;

Parameters and result:

Name	Type	Description
active_OnFirstProgress_Init	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnFirstProgress_Loop	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnFirstProgress_Complete	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the DUDE engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_SecondProgressEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnSecondProgress_Init**;
- **OnSecondProgress_Loop**;
- **OnSecondProgress_Complete**;

Parameters and result:

Name	Type	Description
active_OnSecondProgress_Init	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnSecondProgress_Loop	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnSecondProgress_Complete	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the DUDE engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_CommunicationEvents

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- **OnSendCommand**;
- **OnWait**;
- **OnReceiveAnswer**;
- **OnStatusChange**;
- **OnError**;

Parameters and result:

Name	Type	Description
active_OnSendCommand	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnWait	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnReceiveAnswer	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnStatusChange	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_OnError	Boolean	The value of the parameter activates or deactivates the corresponding event.
save_ToSettings	Boolean	If the value of this parameter is true – the DUDE engine will try to save the changes into its settings file. If the value is not true – the changes take effect only for the current use of the COM server.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

set_BehaviorOnOpenClose

With execution of this method the client applications can activate or deactivate the raising of the fallowing event:

- active_EventOnAfterOpenConnection;
- active_EventOnAfterCloseConnection;
- active_EventOnAfterSettingsChange;

Also related to value of the property:

- save_Settings_AfterOpenConnection;
- register_ActiveObjectOnStart;

Parameters and result:

Name	Type	Description
register_ActiveObjectOnStart	Boolean	By default the value of this setting is false. If the value is true – DUDE will try to register ActiveX object into the Running Object Table (ROT).
save_Settings_AfterOpenConnection	Boolean	The value of the parameter activates or deactivates the corresponding behavior.
active_EventOnAfterOpenConnection	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_EventOnAfterCloseConnection	Boolean	The value of the parameter activates or deactivates the corresponding event.
active_EventOnAfterSettingsChange	Boolean	The value of the parameter activates or deactivates the corresponding event.
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

Services

Operations with files

set_Download_Path

Client application must set the download path for the downloading ANAF files by execution of this method. This method must be executed before execution of some of the methods: "download_ANAF_Zrange" or "download_ANAF_DTRange".

Parameters:

Name	Type	Description
Value	WideString BSTR	The target download path (an existing folder).
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

download_ANAF_Zrange

This method downloads ANAF files from the fiscal device according to the values of: "**zRange_StartValue**", "**zRange_EndValue**", "**download_Path**".

During the execution of the methods DUDE will fire more than one time many of the events, so if the downloading of the ANAF files from client application must be accelerated as a process – the application can deactivate some of the events in the beginning and after execution of the method the application can switch them back on to an active state.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

download_ANAF_DTRange

This method downloads ANAF files from the fiscal device according to the values of: “**DateRange_StartValue**”, “**DateRange_EndValue**”, “**download_Path**”.

During the execution of the methods DUDE will fire more than one time many of the events, so if the downloading of the ANAF files from client application must be accelerated as a process – the application can deactivate some of the events in the beginning and after execution of the method the application can switch them back on to an active state.

Parameters:

Name	Type	Description
Result	Integer / Long	Result is equal to zero if method is successfully executed and less than zero if error occurs. If the result is negative – the value contains the last error code from the DUDE engine.

cancel_Loop

With execution of this method – you can tell the DUDE engine to stop the loop process and after that to exit the current execution of the method. For example: In case of the wrong range parameters – the download process of the ANAF files can be very time-consuming.

Parameters: **None**

Result: **None**

upload_Logo

This method sends a logo file to the fiscal device. The file must be with proper size, monochrome and with “bmp” extension. The fiscal devices not accept images different than bmp.

During the execution of the methods DUDE will fire more than one time many of the events, so if the uploading of the logo files from client application must be accelerated as a process – the application can deactivate some of the events in the beginning. After the execution of the method the application can switch them back on to an active state.

Parameters:

Name	Type	Description
FileName	WideString BSTR	The path and file name to the logo file.
Result	WideString BSTR	A system information according to the input search value.

upload_Stamp

This method sends a stamp image file to the fiscal device. The file must be with proper size, monochrome and with “bmp” extension. The fiscal devices not accept images different than “bmp”.

During the execution of the methods DUDE will fire more than one time many of the events, so if the uploading of the logo files from client application must be accelerated as a process – the application can deactivate some of the events in the beginning. After the execution of the method the application can switch them back on to an active state.

Parameters:

Name	Type	Description
FileName	WideString BSTR	The path and file name to the stamp file.
StampName	WideString BSTR	Name of the stamp as file name into the fiscal device in format 8.3 *Example: "stamp123.bmp".*
Result	WideString BSTR	A system information according to the input search value.

SystemInfo

get_SystemInfoSearchList

This method returns a list of system values which can be used as a search parameter into the method "**get_SystemInfo**". These two methods are suggested for the convenience of programmers.

Parameters:

Name	Type	Description
Result	WideString BSTR	A list of system values which can be used as a search parameter with the method "get_SystemInfo".

get_SystemInfo

The programmers can use this method as it is convenient. It shows information about the current PC configuration, folders and so on in the way that the DUDE sees the system.

Parameters:

Name	Type	Description
SearchValue	WideString BSTR	One of the items returned from the method: "**get_SystemInfoSearchList**".
Result	WideString BSTR	A system information according to the input search value.

Service methods

upload_Certificate

This method sends a certificate file to the fiscal device. **Do not execute this method! It is for service needs only**.

Parameters:

Name	Type	Description
FileName	WideString BSTR	The path and file name to the certificate file.
Result	WideString BSTR	A system information according to the input search value.

upload_Profile

This method sends a profile file to the fiscal device. **Do not execute this method! It is for service needs only**.

Parameters:

Name	Type	Description
FileName	WideString BSTR	The path and file name to the profile file.
Result	WideString BSTR	A system information according to the input search value.

Commands to the fiscal device

execute_Command

Common method for sending commands to the fiscal device. The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError;

Parameters and result:

Name	Type	Description
Command	Integer/Long	The value of the command in the decimal number system.
input_Value	WideString BSTR	The logical data of the command – formatted according to the protocol of the fiscal devices. You can find a detailed description of the commands to the fiscal device in the documents: • In English: FP_Protocol_EN.pdf • In Romanian: FP_Protocol_RO.pdf These documents are located in the folder DOCUMENTATION which is a sub-folder of the root of the installation folder.
output_Value	WideString BSTR	Text which contains the logical data from the answer from the fiscal device. The answer is formatted according to the protocol of the fiscal devices. You can parse the answer with your own parser or use the parsed answer from the property "last_AnswerList".
Result	Boolean	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

execute_Script_V1

Common method for sending a set of commands to the fiscal device. The client application can execute pre-prepared text scripts through this method. The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnBeforeScriptExecute;
- OnScriptRowExecute;
- OnAfterScriptExecute;
- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError;

Parameters and result:

Name	Type	Description
scriptType	TScriptType	This value is used from the script engine. DUDE must know the language of the incoming script. Possible values: • DS • FPrint ***Note:*** *In this version of DUDE - the FPrint script type is not supported.*
input_Value	WideString BSTR	The script text which contains the complette set of needed commands. Format of the text row: <***cmd***>,<***logical data for the command***> • cmd – the value of the command in the decimal number system; • logical data of the command – formatted according to the protocol of the fiscal devices. You can find a detailed description of the commands to the fiscal device in the documents: • In English: FP_Protocol_EN.pdf • In Romanian: FP_Protocol_RO.pdf These documents are located in the folder DOCUMENTATION which is a sub-folder of the root of the installation folder.
Result	Boolean	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_ComandsList

The DUDE engine contains an internal description of the commands for the given fiscal device as a collection of objects and the client applications can use them. This method returns a list with “human names” of the commands from the protocol of the fiscal device. In this version of the DUDE engine – the programmers can choose from three different names for each of the commands and use them according to their own preferences.

Parameters:

Name	Type	Description
Index	Integer/Long	A zero based index for the internal search machine. Use the method with values of index between 0 and 2.
Result	WideString BSTR	This method returns a list of command names accordingly to the used index value.

Format description of the “human names”:

- index [0]: <**cmd**>_<**group_name**>_<**name**>. *Example: 038_receipt_NonFiscal_Open;*
- index [1]: <**group_name**>_<**name**>. *Example: receipt_NonFiscal_Open;*
- index [2]: <**name**>. *Example: NonFiscal_Open;*

In some cases the “human names” for index 1 and 2 can be identical. You can easily find the detailed description for a given command in the documentation of the low level protocol by searching for a given **cmd** value and, of course, the **name** value.

get_CommandInfo

By using this method the client applications can get from the DUDE engine some general information about the command and its parameters. Look at the source code of the demo programs for a demonstration of the method. This method is developed for the programmers as a fast way of getting information about the command and its input or output parameters. Of course – all of these parameters are described in the low level protocol documentation but here you can found information on the limit values of the parameters used by the DUDE engine.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	The one of the "human names" of the given command exported from the method “get_CommandsList”.
Value	WideString BSTR	Text with a common information of the command and its input or output parameters.
Result	Integer/Long	If the result of the function is true – this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_InputParams_Count

Through this method – the programmers can receive the count of the input parameters for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	Integer/Long	The count of the input parameters for a given command.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_InputParams_Names

Through this method – the programmers can receive the list of the input parameter names for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	WideString BSTR	The list of input parameter names accordingly to the used “**command_Name**” value.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_InputParam_ByIndex

If you want to use the method "**execute_Command_ByName**" you must ensure that all input parameters of the command are set properly before the execution of the command. One of the ways to do that is to use this method.

Notes:

- *In some cases the value of the input parameter must be empty. For example, look at command 255 in the case of reading of the value from fiscal device.*
- *In other cases the input parameter is optional – so depending on the case, the input parameter may or may not need to be empty.*
- ***You must be sure that you set the values of all input parameters properly. The DUDE engine keeps the input parameters into the memory so in some cases you probably do not want to execute the method with no set new values (using the old values).***

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Index	Integer/Long	A zero based index related to the input parameters list of this command.
Value	WideString BSTR	The value of the input parameter. Check the low level protocol to be sure that you will fill the value of this parameter properly.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

set_InputParam_ByName

If you want to use the method "**execute_Command_ByName**" you must ensure that all input parameters of the command are set properly before the execution of the command. One of the ways to do that is to use this method.

Notes:

- *In some cases the value of the input parameter must be empty. For example look at command 255 in the case of reading of the value from fiscal device.*
- *In other cases the input parameter is optional – so according to the case, the input parameter may or may not need to be empty.*
- ***You must be sure that you set the values of all input parameters properly. The DUDE engine keeps the input parameters in the memory so in some cases you probably do not want to execute the method with no set new values (using the old values).***

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Name	WideString BSTR	The input parameter name.
Value	WideString BSTR	The value of the input parameter. Check the low level protocol to be sure that you will fill the value of this parameter properly.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

execute_Command_ByName

The DUDE engine contains an internal description of the commands for the given fiscal device as a collection of objects and the client applications can use them. Set the input parameters of the command properly – before the execution of the method.

The execution of the command can fire one or more of the following events if they are not switched off programmatically:

- OnSendCommand;
- OnWait;
- OnReceiveAnswer;
- OnStatusChange;
- OnError*;*

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

After the successful execution of the method – you can receive the output parameters by using one of the following methods:

- get_OutputParam_ByIndex;
- get_OutputParam_ByName;
- use the values from last_AnswerList;

Note: Any of the commands exported into the "human oriented" command names list can be executed with your own input text via method “execute_Command”. You can execute each of the commands from the low level protocol of the fiscal device via method “execute_Command”. During the development of the client application ***you have more than one way to execute the commands*** *if:*

- *something goes wrong;*
- *something is not clear;*
- *the command is not exported for the execution by name;*

get_OutputParams_Count

Through this method – the programmers can receive the count of the output parameters for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
Value	Integer/Long	The count of the output parameters for a given command.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParams_Names

Through this method – the programmers can receive the list of the output parameter names for a given command.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	The one of the "human names" of the given command exported from the method “**get_CommandsList**”.
Value	WideString BSTR	The list of output parameter names accordingly to the used “**command_Name**” value.
Result	Integer/Long	If the result of the function is true - this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParam_ByIndex

After the successful execution of the method “**execute_Command_ByName**” – you can receive the output parameters by using this method.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Index	Integer/Long	A zero based index related to the output parameters list of this command.
param_Value	WideString BSTR	The value of the output parameter.
Result	Integer/Long	If the result of the function is true – this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

get_OutputParam_ByName

After the successful execution of the method “**execute_Command_ByName**” – you can receive the output parameters by using this method.

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three “human names” possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
param_Name	WideString BSTR	The output parameter name.
param_Value	WideString BSTR	The value of the output parameter.
Result	Integer/Long	If the result of the function is true – this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

generate_SourceCode

In this version of the DUDE engine – the programmers can use this method to generate a source code which gives an example of the usage of method "**execute_Command_ByName**". The source code can be used and modified as is comfortable for the programmer and without any license restrictions. In simple terms – it is free as a beer. Of course, this source code is a generated from our AI example so if you choose to use it – please ensure that it is implemented and works well on a real fiscal device.

Currently the DUDE engine can generate a source code on two languages for all commands which are exported from the method "**get_ComandsList**".

Parameters:

Name	Type	Description
command_Name	WideString BSTR	Use one of the three "human names" possibilities received by the command "**get_CommandsList**" or from the document "**CommandsList.xls**".
code_Type	TCodeType	Possible parameter values: • Delphi; • CSharp;
Result	Integer/Long	If the result of the function is true – this mean that the DUDE engine internally checks the state of the given status bit and if it is raised in the answer of some command – the engine will return an error as a result of the command execution.

***Note**: In this version of DUDE – the driver generate source code only on Pascal (Delphi) and C#. If you find a bug it would be nice to send an email to the authors of the COM server. We would be glad to fix or resolve the issue in the next version of the engine.*

How to send a bug report

To resolve the issue – we must simulate the same conditions as yours. Please send us an email with the fallowing information:

- ✔ The version of the DUDE;
- ✔ The information about the Windows and about system – you can use the information returned from the method “get_SystemInfo” or from the demo program;
- ✔ The information about the model of the fiscal device. It would be nice and if you print and scan the diagnostic information from the device;
- ✔ The type of the transport protocol and if you use an adapter for USB to RS-232 – the model of the adapter;
- ✔ If you use the method “execute_Command” - please send us the value of the command, the value of the input parameter and the output if exist;
- ✔ If you use the method “execute_Script_V1” - please send us the value if the script;
- ✔ If it is possible:
  - activate the tracking mode to true;
  - simulate the problem step by step;
  - send us the log file;
- ✔ If it is possible – save and send to us the values of the status bit properties after execution of the failed command. Do not execute other command after failed attempt – just get the values and save them;
- ✔ Send us the value of error code;
- ✔ Send us an information about your programming environment;
- ✔ If you use a generated code (or your own) and the method “execute_Command_ByName”, please send us all needed information – we will try to simulate the same conditions.
- ✔ Always send us the values of the input parameters;
- ✔ **If you will want to send us a source code – please do not send the entire project!** We do not offer such support!

Our support is only by emails!

Please do not ask us for a phone, SkyPe, Viber and any different types of the conference calls.

We know that your project is a very important to you and please believe us – your success is important for us too. We will be glad to help you but the policy of the company is to help programmers only via emails due to the specificity of the topics being discussed.

Contacts

If you find a bug – please send us an email. We would be glad to fix or resolve the issue in the next version of the DUDE. Also, if you are a programmer – you are free to ask us about the different aspects of the usage of the server or for any other problems or recommendations related to the DUDE. We will try to help you.

You can send the emails with your questions to:

suport.datecs@danubius-exim.ro

dobrin@datecs.bg