ACDQ-CDA
CCD (Common Communication Driver)

Installation and programming guide

version : V3.2.1 October 2010

Revisions			
Version	**Date**	**Author**	**Description**
1.0	1997-10-23	Farid Agharabi	Original version
1.1b	April 98	FA	Added new parameters and CDA version 3.0 support
1.9f	Dec 2000	FA	New parameters added to CCD.INI File locking bug fix. New error codes to explain the infamous 1031 protocol error.
1.9i	April 2001	FA	Adding an activity file (for windows version). See section 3. Correcting the timeout problem error 1002 on some type of modems. Updating MS-DOS documentation.
2.0	Nov 2001	FA	Windows Telephony API (TAPI) support ccd.ini in Windows ini format Supporting variables and Windows environment variables inside CCD.INI file.
2.1	Jan 2002	FA	Added options for converting French characters in V2 and parameters for adjusting the number of redial attempts and the delays between each attempt. Verification of illegal characters (control characters) and accented characters in the input file.
3.1	October 2007	FA	Updating crypto info IP based transaction, Redirection file, supplemental ini file
3.2	October 2010	FA	Capacity to check for updates Capacity to be started as a Service

The Common Communication Driver has been developped for
"Association des chirurgiens dentistes du Québec" and the "Canadian Dental Association"
by
Stratégica 2000 Informatique inc.

Contents

Introduction

The ACDQ-CDAnet common communication protocol and standard has been designed in order to present a simple and unified way of transmitting dental claims through the ACDQ-CDAnet network.
The ACDQ-CDAnet Common Communication Driver or "CCD" is the client-side implementation of the protocol on various PC platforms.
This document is primarily intended to help software vendors to install and configure the CCD. The document also explains the mechanisms through which the dental office software should interact with the CCD.

Key features

- The CCD implements the ACDQ-CDAnet common communication protocol and can transmit CDAnet version 2.5, 3.0 and ACDQ-CDAnet version 4.0 transactions to every network that accepts this protocol;
- The CCD also supports the compression and encryption methods specified by the ACDQ-CDAnet version 4.0;
- The CCD is available Windows 32 platforms;
- The CCD supports various connection methods: direct asynchronous serial link, dialup link, Datapac 3101 and dedicated Datapac[1]; TCP/IP
- The CCD includes a connection script language and a set of parameters for each network. This will let the CDD easily adapt to each PC/Network configuration for using direct and dialup connections and logging procedures;
-
- The CCD uses a simple directory and file scheme for interfacing with the dental office software and can be used on a single PC or a networked PC used as gateway;
- The CCD Can use a redirection table in order to make it easy for a carrier transaction to be processed by a different CDAnet Network without having to re-configure the dental software;
- The CCD can update itself, (if the CCDUPDATE program is installed and configured)
- The CCD can be started as a Windows service
- Finally, the CCD includes extended log and trace file and an internal test facility for developing and debugging the interface with your software.

System Requirements

- Microsoft Windows 2000/XP/7 (not tested on Vista)
- 2 Mb of disk space
- a free serial COM port or USB modem or Network Interface Card

[1] A powerful connection script allows for the use of virtually any character oriented connection method as long as it can be accessed through the PC's serial port.

- an optional Hayes compatible modem (for dialup connection)

. Getting Started

Before installing the CCD

Make sure that the COM port does not conflict with another device.
If you are using a modem, make sure it is installed and operating properly. USB modems should have their latest driver, for the TAPI (Telephony Application Programming Interface) option to work properly.

Installing the CCD

Obtain the latest version of the CCD (for example CCD32-3.2.1.zip) and unzip its content in a directory [C:\CCD]
The following table summarizes the files that you should install in the CCD directory:

OS	Remark
Windows	CCD32.EXE Main program (32-bits) CCD.INI initialization file ACDQ-CDA.CFG Encryption key file

Running the CCD

The CCD can be run from the command prompt. Please use the appropriate version of the program according to your operating system.
CCD32
example:
C:\CCD\CCD32 -v
Use the command line option "-v" or the menu item "Help/detailed messages" for increasing the level of log messages displayed and saved into the log file. This will help you troubleshoot your installation.
Option -x disables File/Exit menu and Alt-F4. In this case you can only stop the program via the task manager.
Note: **in Windows, only a single instance of the program can run** at a given time, so restarting the CCD will not start a new instance. If it is required to do so, for example to use a second modem simultaneously, you will have to start the second CCD from a another directory (for example in C:\CCD2 directory)

Windows NT/2000 Terminal server

The CCD recognizes the terminal server environment and starts a single instance of the program.

Windows in a network environment

The CCD will not accept to run in a directory where another CCD is currently active. For example in a network and a shared file, only one CCD should be started.

Multiple users session active on the same computer

The CCD will not accept to run in a directory where another CCD is currently active. Users on different sessions must start the CCD in different directories. It is a better idea to start the CCD as a Windows service so that it can process transactions for all user sessions.

Running the CCD as a service

Since version 3.2, the CCD can be started as a Windows service. The Service can be created with the following command line:

```
sc create CCD binPath= "C:\CCD\CCD32.exe -s" start= demand
displayName= ".CCD ACDQ Common Communication Driver"
```

- Note the command line option **-s** that instructs the CCD to start as a service
- Specify the binPath argument as the full path of the ccd32.exe program.

The Windows service can be configured to start when Windows starts.
It can also be started from the command line.

```
sc start CCD
```

When the CCD is started as a service, the CCD.log file should be used to verify that the startup is successful.

```
INFO - CCD service: Starting ... ExeDir='C:\CCD\CCD32'
INFO - Common Communication Driver V3.2.1, 2011-10-14, Copyright ACDQ 2011
INFO - CCD service: starting main CCD thread
INFO - CCD service: Service is running.
```

> The service must be named "CCD" otherwise the program will not be able to restart itself after an update.

. Configuration

The configuration of the CCD is stored in an ASCII file (by default CCD.INI). This file can be edited by using any text editor.
The configuration file has the same syntax across all supported platforms.
Since version 2.0, the CCD supports the Windows ini format so the ccd.ini file entries can me modified using Windows API.
The old ini format is still supported but should not be mixed with the Windows format.
Since version 3.1 the CCD supports a second supplemental initialization file (ccdsup.ini) that allows additional network configuration to be added without having to modify the original ccd.ini. This can be used for adding services without disturbing the particular setup defined in ccd.ini. The supplemental initialization file contains network definitions with the same parameters and format than ccd.ini file.

CCD.INI format and syntax

The ini file is composed of three sections:

❶
```
general begin
  ... ;  general parameters
end
```

❷
```
connectScript _s1 = {...} ; script _s1
connectScript _s2 = {...} ; script s2
```

❸
```
network XYZ
begin
  primary connection begin
    ... ;  primary connection parameters
  end
  secondary connection begin
    ... ;  secondary connection parameters
  end
end
```

Alternate Windows ini format: (supported since version 2.0)

❶
```
[general]
;  general parameters
params=value
$variables=”value”
```

❷
```
[connectScript]
; script _s1
_s1 = {...}
; script s2
_s2 = {...}
```

❸
```
; network XYZ definition primary
[XYZ]
params=value

; network XYZ definition (secondary or backup connection)
[SECONDARY_XYZ]
params=value
```

❶ The general parameter section	This section contains the parameters that are common to all networks that can be accessed through the CCD. For example the COM port. This section can also contain variable definitions in the form $varname=”string”. These variables can be used in network sections in place of value strings for phone and address parameters.
❷ The script section	This section contains one or more scripts that can be used for special connections that require a logging procedure. For example Datapac.
❸ The networks section	This section contains one or more network subsections describing the parameters particular to each network.

The following text describes in detail the content of each section.

General parameters section

defcomdev = “COM1” | “COM2” | “COM3” | “COM4” | “ COMn” | TAPI | TAPIn | “exact tapi device name”

Default communication port (modem port for dialup connections). Make sure that the COM port is configured properly and uses the standard Base I/O Port Address and IRQ setting.

COM Port Setting	COM1	COM2	COM3	COM4
Base I/O Port Address	3F8	2F8	3E8	2E8
IRQ	IRQ4	IRQ3	IRQ4	IRQ3

example:
defcomdev = “COM2”
Note: in Windows every COM port name recognized by Windows is acceptable.
In all WIN32 platforms, since CCD version 2.0, you can use TAPI as a valid comdev name. In this case the program looks for the first data modem capable device installed in Windows. TAPI2 will select the second data capable device and so on.
Alternately you can specify the exact TAPI device name:
For example:
defcomdev = “U.S. Robotics 56K FAX EXT pnp”
Look in Windows/Control Panel/Modems for the list of TAPI modems installed in your computer.

> Since CCD version 3.0 you can specify **connectiontype=tcpip**, for a given network

in this case defcomdev parameter is ignored.

modemInit = string

Default modem initialization string containing a series of commands that prepare the modem for the type of dialup connection.
This string should always contain the "ATHE0V1Q0" commands followed by additional commands as required.
This parameter is ignored for direct connections and TAPI modems
Example: modemInit = “ATHE0V1Q0”

HSmodemInit = string

Same as modemInit but used for high speed connections (when the baud rate is >9600)

modemHangup = string

This command is sent to the modem for forcing it to hang-up. It should contain “ATH” followed by additional commands if you need to put the modem into a predetermined state after disconnecting the line.
This parameter is ignored for direct connections.
Example: modemHangup = “ATH”
Note: when the modem is on-line the CCD send the default escape code (“+++”) sequence in order to bring the modem into command line mode. Make sure that the modem is setup properly.
This parameter is ignored for TAPI connections.

inputFileName = name

The inputFileName parameter is the name of the file containing the request (dental claim).
Please follow the following syntax:

- Use only characters in [A-Z] $ and _ range
- DO NOT specify the extension of the file as the CCD looks for all extensions “.*”
- DO NOT specify more that 8 characters even if your operating system allows you to create long file names.

Example: inputFileName = INPUT
In the example the CCD looks for all file named INPUT.*
The **file is locked by the CCD during the transfer**, Please read the section explaining the interface between the dental office software and the CCD (section 3).

outputFileName = name

The outputFileName parameter is the name of the file that will contain the result of the transaction of any error code returned by the processor or the CCD.

Please follow the following syntax:

- Use only characters in [A-Z] $ and _ range
- DO NOT specify the extension of the file as the CCD creates the same extension that the input file created by the dental office software.
- DO NOT specify more that 8 characters even if your operating system allows you to create long file names.

Example: outputFileName = OUTPUT
In the example the CCD creates the file OUTPUT.xxx in the network's subdirectory. The extension will be the same as the input file.
The interface between the dental office software and the CCD is described in section 3.

hangAround = TRUE | FALSE

If set to TRUE, the CCD will process all current transactions then it will wait indefinitely for a new transaction (input file) to process. Otherwise, the CCD will disconnect the COM port and exit. In the later case it is necessary to restart the CCD for handling each new transaction.

logFile = string

This parameter contains the name of the file that will contain the log of each transaction. This file is erased after each startup.
This parameter is ignored if logging is not enabled.

loggingEnabled = TRUE | FALSE

Enables or disables the logging of transactions into the file (see parameter **logFile**).

logFileAppend = TRUE | FALSE

This is an optional parameter, if true, the CCD will append new messages at the end of the current log file. If the parameter is absent or set to "false", and new log file is created every time the application starts.

detailedMessages = TRUE | FALSE

This is an optional parameter equivalent to the –v switch of the command line. It activates the generation of more detailed messages in the log file, useful for investigating connection problems.
If this option is set to FALSE in the ccd.ini file it can still be overridden by the command line switch –v.
The default value is FALSE.

v2convertAccent = TRUE | FALSE

This is an optional parameter. If set to TRUE, it activates the conversion of accented characters in the INPUT.xxx file into the non-accented characters.

The conversion occurs only for CDAnet V2 (and V3) transactions and only on the input file. The carrier response is never converted.
The CCD assumes that the accented characters in the input file are coded using Code Page 850 as defined in CDANet v4.0 Format and Standards.
The default value is TRUE
*Note: This parameter can also appear in a **network connection bloc** to override the general behavior.*

dialRetries = integer

This is an optional parameter, which sets the number of times that the CCD will attempt to redial the network connection after a failure (e.g. busy signal)
The default value is 2 retries.
*Note: This parameter can also appear in a **network connection bloc** to override the general behavior.*

delayBetweenRetries = integer

This is an optional parameter, which sets the delay between each redial attempt .
The default value is 2 seconds
*Note: This parameter can also appear in a **network connection bloc** to override the general behavior.*

disconnectAfterTX = TRUE | FALSE

This option is useful in combination with the hangAround parameter, when it is necessary to disconnect the COM port after each transaction so another application can use it.
If a new transaction must be processed but the COM port is not available, the CCD will return an appropriate error code in the output file.

hangupDelay = integer

This option is used for specifying the number of seconds (between 0 and 99) that the CCD will wait before hanging up the phone line, after the last transaction is sent.
For multi-paged or batch transactions, this will let the dental office software process the last response and create the next request file. If the destination of this request is the same network as the precedent, the request will be sent during the same session without having to hang up and dial the same network again.
Note that the dental office software is responsible for checking the status of each transaction before sending the next. For example, if the first page of a multi-paged transaction is rejected, the following pages should not be sent.

Variable definition $variable=”value”

The general section can contain variable definitions, a variable can only have a string value and is useful for keeping the phone numbers or datapac addresses in the same section for easy manipulation.

Variable names are not case sensitive, a variable definition starts with a '$' followed by the variable name, the variable name cannot have an embedded space.
Exemple
$ThisTownDatapac="555-200-2100"
…
[XYZ]
…
phone = $(thisTownDatapac)

IMPORTANT NOTE: If the variable is not defined in the ccd.ini file, the CCD tries to look for an environment variable with the same name.
Example:
C:\CCD\> SET ThisTownDatapac ="555-200-2100"
TIP: You can use this method in order to customize each installation to use a different phone number without having to modify the ccd.ini. Set the correct datapac phone numbers in the environment variable table (either in AUTOEXEC.BAT or using the Windows Environment Variables menu) and use the variable in the ccd.ini .

Connection scripts section

This section of the ini file can contain zero or more connection scripts. Each script has a unique name and can be used to setup a particular connection procedure after the modem connects to the remote claim processor.
In case of a direct connection, the script is executed immediately after the COM port is openned.

<_name> = { <script statements> }

The script name must be unique (e.g. you cannot re-define a script), it must start with the underscore character "_".
The script statements are composed of one or more instructions between two curly brackets "{" "}".
In this example, we are defining a script named _datapac.

```
connectScript _datapac  ={ transmit "."
                           wait 3 for "DATAPAC:" then {
                             transmit address "^M"
                             wait 3 for "connected" then{
                               return 0
                             }
                           }
                         }
```

If Windows ini format is used all scripts must be in the same section, and the script should be kept on a single line with no carriage returns.

```
[ConnectScript]
_datapac={transmit "." wait 3 for "DATAPAC:" then { transmit address ^M"... return 0}
_xyz={transmit ".." wait 3 return 0 }
```

Executing scripts

After successfully connecting to the network, the CCD automatically executes the connection script specified for that network.
Each script should contain at least one [return n] statement. As a general rule, a script is considered successful when it returns with the value zero. If the script returns another value or if it ends by reaching the last curly bracket "}". The CCD will consider it a failure to connect and will disconnect the modem and return an appropriate error code in the output file.
If the script is successful the CCD will wait for the processor request to transmit signal which is an <ENQ> (dec 5) character. *It is not necessary to incorporate a test for this character into the script.*

Passing parameters to scripts

Currently there is no way to pass external parameters to a script. With the only exception of the "address" field of a network which can be sent using the [transmit address "^M"] instruction.

> **Note:** The exact syntax of all script instructions is explained in the appendix.

Network parameters section

The CCD can be configured to dial more than one network. This section of the ini file is where all parameters regarding each supported network are defined.
Each network is defined within a network definition Bloc.

network <name> begin ... end

alternate win ini syntax **[networkname] …**

This is a network definition bloc.
A network definition bloc is used for defining all parameters regarding a named network.
The network name must be unique e.g. you cannot define a network more than once.
The formal syntax is:

```
network <networkName>
begin
  [prefix = string | default]
  [maxProcessingDelay = n]
  [maxEnqDelay = n]
  [disconnectMethod = soft|hard|both]
  certificateStore = CURRENT_USER:MY|LOCAL_MACHINE:MY|file:ccd.cer
  certificateNetworkId="certificate name for this network"
  certificateCCDId=default|provider|clinic|none|"string"
 primary connection
  begin
   ...
  end
  [secondary connection
   begin
     ...
   end]
end
```

Or If Widows ini syntax is required:

```
[networkName]
prefix = string | default
maxProcessingDelay = n
maxEnqDelay = n
disconnectMethod = soft|hard|both
connection parameters
...

[secondary_networkname]
prefix = string | default
maxProcessingDelay = n
maxEnqDelay = n
disconnectMethod = soft|hard|both
connection parameters
...
```

The following sections define the network parameters.

prefix = string | default

This is parameter is required by some networks to identify dental transactions. So each network can have a different prefix.. If the parameter is not specified or set to default, the value “11nn1111” will be used, where nn represents the CCD version (e.g 19 for v1.9) The string replaces the CDAnet A01 field of the input transaction and is usually returned intact in the response transaction.

maxProcessingDelay = n

The maximum delay the CCD will wait for the network to process a claim, this is an optional parameter, and the default is 30 seconds. If no response is received within that period, the CCD will return a 1042 status code (server timeout)

maxEnqDelay = n

The maximum delay (in seconds) the CCD will wait for the network's prompt (ENQ) to send the claim, after the connection script is successful. This is an optional parameter, the default is 30 seconds.
If the ENQ is not received, the CCD will return a 1041 status code (ENQ timeout)

disconnectMethod = soft|hard|both

This parameter controls the way the modem will hang-up. The soft method sends a "+++" followed by "modemHangup" string (usually ATH). The hard method lowers the RS-232 signal line DTR for one second, which is a very quick way to disconnect the line, if the modem is properly configured (see your manufacturer documentation for more details on modem configuration commands and dip switches)
This is an optional parameter the default is soft.

certificateStore=CURRENT_USER:MY|LOCAL_MACHINE:MY|file:ccd.cer

This parameter points to the certificate store. The CCD uses this store to locate the Network certificate (to get the network's public key).

- **CURRENT_USER:MY** the certificate store is specific to the current logged user.
- **LOCAL_MACHINE:MY** the certificate store is installed on the current machine (same store for all users of this machine)
- **file:ccd.cer** the certifcate store is file based (the file ccd.cer contains the certificate)

certificateNetworkId="string"

This is the certificate ID of the network. The CCD looks for the certificate with that name in the certificate store (see above) and encrypts the request with the Network's public key. The network decrypts the request

certificateCCDId=default| clinic | provider| |none|"string"

The CCD uses this parameter to set the encryption certificate according to some code that uniquely identifies the source of the transaction.

- **default| clinic** : the CCD extracts from the CDAnet request and uses the clinic identification (4 alphanumeric field B02 of CDAnet transaction)
- **provider** : the CCD extracts the provider ID from the CDAnet request and uses it for encryption.
- **"string"** : use the certificate with that name for encryption. The Network should have access to the public key , otherwise it will not be able to decrypt the data sent by the CCD.
- **none:**

<primary|secondary> connection begin ... end

This is connection definition bloc.

Each network can have a primary and secondary (backup) connection. The primary connection must always be specified, while the secondary connection is optional.
If the network cannot be reached after two attempts, the CCD will try the secondary connection.
The formal syntax is

```
primary connection begin
  ...; primary connection parameters
end
```

Important Note: The primary or backup connection definition MUST appear inside a network definition bloc.

Windows ini syntax note:

In windows ini syntax the primary connection bloc parameters are located in the main network section for examples section [xyz] for network xyz. The secondary (backup) connection definitions are located in another ini section called [secondary_xyz]

dataBit = 7 | 8

The 'dataBit' field is a member of the connection definition bloc. This field specifies the length of serial data bits.

> Note that the compression and encryption of the CDAnet version 4.0 requires an 8 bit, no parity connection.

Before adding a new network, make sure that it can handle 8-bit data transfer. Some networks can have different addresses (DNA) or phone numbers for handling 7-bit and 8-bit data.

parity = none | even | odd

The 'parity' field is a member of the connection definition bloc. This field specifies the data parity of the asynchronous link.

> Note that the compression and encryption of the CDAnet version 4.0 requires an 8 bit, no parity connection.

speed = 1200 | 2400 | 4800 | 9600

The 'speed' field is a member of the connection definition bloc. This field specifies the connection speed of the asynchronous link.

phone = string

The 'phone' field is a member of the connection definition bloc. This field specifies the network's phone number for the given connection definition bloc (primary or secondary). If the dental office phone is connected to a pulse activated phone line, add a 'P' at the beginning of the phone number.
Example:

```
....
;  phone =”9,123-4566” for dialing 9 before calling
phone =”123-4566”
...
; use the value of variable simPhone defined in general section
phone = $(simPhone)
```

address = string

If connectionType is "tcpip", the *address* parameter is the IP address and port number of a switch capable of receiving CDAnet transactions. For example Net+ switch.
The syntax is "ip-address or hosr name" : "ip-port-number"
Example:

```
...
 address  = "switch.cdanet.ca:7898"
...
```

For default mode connection using datapac,
The ‘address’ field is a member of the connection definition bloc. This field specifies the network’s address (DNA) for the given connection definition bloc (primary or secondary).
The address is used for connections that require a DNA after dialing the phone number. For example DATPAC address.
Note that the script command ‘transmit’ can send this field when the following syntax is used.

```
connectionScript ={
     transmit address “^M”
}
```

Example:

```
...
 address  = “8887662566”
...
```

compress= TRUE | FALSE

The ‘compress’ field is a member of the connection definition bloc. This field activates or deactivates the network’s compression mode for a given connection definition bloc.
You should leave the compress=TRUE for all CDAnet version 4.0 transmissions.

encrypt =TRUE | FALSE

The ‘encrypt’ field is a member of the connection definition bloc. This field activates or deactivates the network’s encryption mode for a given connection definition bloc.
You should leave the encrypt=TRUE for all CDAnet version 4.0. The encryption is also controlled by the field A10 of the CDAnet version 4.0 requests. An attempt is made to encrypt a transaction when the field A10 of your request is set to “2” or “3”.
However, the CCD will not be able to transmit encrypted messages in the following cases:

- The ‘encrypt’ field is set to FALSE;
- Or the dataBit field for this connection bloc is set to 7 (encryption uses 8-bit characters);

- Or the encryption key file (CDA-ACDQ.CFG) is invalid or missing.

An appropriate error status will be written into the output file.

connectionMode = dialup | direct

The 'connectionMode' field is a member of the connection definition bloc.
The CCD selects a different connection mode according to the settings of the field:

- When set to 'dialup', the CCD will attempt to use a connected modem in order to establish a dialup communication with the remote host. The network's phone number should be specified and the modem initialization strings must be valid.
- When set to 'direct', the CCD will consider that the network is directly connect to the serial port, via a dedicated serial line.
- When **connectionType** = tcpip the connectionMode = direct is assumed;

This field does not affect the connection script, which is executed after a successful connection to the dedicated line, or the remote modem.

connectionType = tcpip | datapac

The 'connectionType' field is a member of the connection definition bloc.
Since CCD version 3.0 connectionType=tcpip can specified in order to specify a network communication parameter bloc for a TCP/IP connected CDAnet Switch that supports CDAnet Transaction. (For example Net+)
When ***connectionType***=tcpip:

- the ***address*** parameter must be defined as the CDAnet Swicth Hostname and port number
- the ***connectionMode*** is assumed to be direct
- certificateStore, certificateNetworkId, certificateCCDId parameters must be assigned according to the configuration provided by the CDAnet Switch. This
- comdev, databit, parity and speed parameters are ignored, and default Network Interface Card is used.

connectionTimeout = <number_of_seconds>

This field specifies the maximum amount of time that the CCD should wait before it receives the "CONNECT" message from the modem.

dialRetries = integer

Note: This parameter overrides the default value or the value defined in the general section (see General parameter section).

delayBetweenRetries = integer

Note: This parameter overrides the default value or the value defined in the general section (see General parameter section).

disconnectAfterTX = TRUE | FALSE

Note: This parameter overrides the default value or the value defined in the general section (see General parameter section).

connectScript = use <name> | { <script statements...> }

This field specifies the script that is to be executed after the CCD successfully connects to the serial port (or the remote modem for dialup links).
You can use a script previously defined in the connection script section of the initialization file. You can also define a new script, by using the connection script language defined in the appendix.
Example:

```
...
; uses a script named "_datapac"
connectScript = use _datapac
...
connectScript = { wait 3 for "login:" then
                  {
                    transmit "usrnam^M"
                    wait 10 for "accepted" then return 0
                  }
                  return 999  ; script unsuccessful
                }
```

. Interfacing with the CCD

Introduction

For each configured CDAnet network, there must be a subdirectory in the CCD main directory having the name of that CDAnet network. The same name must also be defined in the CCD initialization file in the CDAnet network definition section.
The dental office software creates the input file (dental claim) in the desired CDAnet network subdirectory.
The name of this file must be the same as the value of parameter "inputFileName". The extension of this file is a 3 character (LUN) Logical Unit Number. The CCD will create the response file in the same subdirectory. The name of this file is defined by the parameter "outputFileName". The extension is the same as the input file.
For example, in order to send a transaction to DENTAIDE:

1. Configure and run the CCD, make sure it runs in the \CCD directory
2. Delete any older output file that may remain \CCD\DENTAIDE\OUTPUT.000
3. Create a file named \CCD\DENTAIDE\INPUT.000

4. Wait for the response file :\CCD\DENTAIDE\OUTPUT.000
5. For sending another transaction go to step 2
6. Stop the CCD and examine the content of \CCD\CCD.LOG see trouble shooting section.

The input file is kept open until the response comes back from the network, so no other process can delete or modify the file during that period.

During the transmission and, the CCD regularly updates the activity file, the dental software can use this feature to monitor the activity and inform the user in the software graphical user interface or by any other appropriate means. See Activity monitoring (activity.dat).

Network environment setup

The network setup is similar to a single workstation setup. A server acts as a CCD server, and usually it is also the file server. The server shares its CCD directory with the other PC's in the network.
Each PC creates an input file in this shared directory, and then waits for the result in the same directory.
In order to prevent one PC to overwrite another PC's INPUT.xxx file, each PC must be assigned with a unique LUN (Logical Unit Number) (e.g. 000, 001, 002, etc.)
For example:
PC 001 creates N:\CCD\BCE\INPUT.001
PC 002 creates N:\CCD\BCE\INPUT.002
The CCD handles the input files one at a time. The result for PC 001 is created in N:\CCD\BCE\OUTPUT.001 and so on.

Testing your setup

The following procedure can be used to understand the concept of interfacing with the CCD and to test your setup.

1. The originally delivered CCD.INI file contains the network names TEST and DENTAIDE; the inputFile parameter is 'INPUT, and the outputFile parameter is set to OUTPUT.
2. Create the subdirectory C:\CCD\TEST;
3. Run the CCD main program, with the working directory set to C:\CCD, it should start and wait for a request;
4. Create a dental claim and rename it to C:\CCD\TEST\INPUT.000;
5. The CCD should read the INPUT.000 file and simulate a connection and return a response in OUTPUT.000

Activity monitoring (activity.dat)

You can monitor the CCD (even if it is running on another PC) by watching the content of the activity file.
The activity file is an ASCII file created in the working directory of the CCD. When the file is present, it means that the CCD is running. This file is destroyed when the CCD terminates.

Furthermore, when the CCD is processing an input file, the content of the file is updated every second so you can monitor the activity of the CCD and inform the user (perhaps by displaying a progress bar).

Activity file format

Note: the CCD holds this file open at all times and your software should open the file in read-only mode. The activity file contains a single record with the following structure.

Field	format	Description
State	Byte	State of CCD - ie. idle, dialing, sending etc. See table 1
Counter	Byte	loops from 32 to 128 (printable characters) updated every second, (only when the CCD is processing an input file)
Network	Character x32	null terminated string network name: being serviced, blank if none
File	Character x32	null terminated string: file name currently being processed (blank if none)
Status	Character x5	status of the last transmission 0=success otherwise see table 2 (status codes)
Computer Name	Character x 16	The name of the computer that runs the CCD

Table 1- CCD State

State name	Code (Decimal)	Description
IDLE	65	Idle waiting for a new file to transmit
READ	66	Found a new file to send (reading)
CONNECT	67	Connecting to modem (serial port)
INIT_MODEM	68	initializing modem
DIAL	69	dialing the phone
EXEC_SCRIPT	70	executing the connection script
ENQ_WAIT	71	waiting for network's prompt to start transmission
SEND	72	sending the request
ACK_WAIT	73	waiting for network's acknowledge
RESP_WAIT	74	waiting for network's response (network is processing)
RECEIVE	75	receiving network's response
LRC_WAIT	76	wait for network's LRC
SEND_ACK	77	send confirmation to network
SEND_EOT	78	send confirmation and end of transmit to network
WRITE	79	writing the response in output file
IDLE_CONNECTED	80	CCD is idle but still connected to the network
HANGUP	81	hanging up the phone line
DISCONNECT	82	disconnecting from serial port

Input file format

The input file is composed of a single CDAnet dental claim, on a single line starting at the first character of the line, with no embedded new-line characters.
The CCD reads the first line (up to the end of line). The remaining data is ignored.
Example:

```
PREFIX00    0009810401666666TS1200213...          ...<CR>
^           ^     ^ ^ ^     ^  ^^    ^
1           2     3 4 5     6  78    9 remaining of the claim

1:A01 prefix
2:A02 Office sequence   (=000981)
3:A03 Format version    (=04)
4:A04 Transaction code  (=01)
5:A05 Carrier Id        (=666666)
6:A06 Vendor Id         (=”TS1”)
7:A10 Encryption Method (=2)   (1=no encryption)
8:A07 Message length    (=00213) length of uncompressed message
```

Note:

- The input file cannot contain any control characters (ASCII code <32),
- The Length field (A07) must be set correctly (e.g. equal to the length of the transaction)
- V2 transaction cannot contain accented (8-bit) characters and are converted to non-accented equivalent (see parameter v2ConvertAccent)

Any anomaly results in the rejection of the transaction with error code 1034.

Output file format

The output file is composed of several comma-separated fields followed by a single CDAnet dental response, on the same line.
The first field, is the sequence number of the request (field A02)
The second field is the status of the transaction (see the status code table)
The third field is the extension of the input file, (logical unit Id)
The remaining is the carrier's response, (only when the transaction could be sent to the network).
Example:

```
981,0,000,...          ...<CR>
^   ^ ^   ^
1   2 3   4 remaining of the claim

1:copy of the field 'Office sequence' from the input file
2:status of the transaction, 0=success (see status code table)
3:Input File extension (logical unit Id)  =000
4:when status=0, the remaining data on this line is the carrier's
response to your claim.
```

Connection status codes

The following table lists the various status codes that can be returned in the output file. The status code is the second field of the output file generated by the CCD. (see Output file format for more details)

Table 2- CCD status codes

Status	Code	Remark
0	success	This code means that the request was sent to the remote host and the response was saved in the output file.
1001	error	A general or Internal error occurred, see log file for more detail.
1002	timeout	This is an unlikely error condition generated by modem incompatibility, or the modem being turned of in the middle of a transaction. see protocol error or dial error.
1003	interrupted	The program or the call was interrupted by the user, or by a system shutdown (win95/NT)
1011	encryption not allowed	The connection definition section parameter 'encrypt' is set to FALSE, but the claim must be encrypted according to field A10 of CDAnet version 4.0 transactions.
1012	cannot encrypt	The connection uses 7-bit characters. CDAnet version 4.0 compression, encryption and the use of French characters require 8-bit connections.
1013	encrypt keyerr	The encryption key file ACDQ-CDA.CFG is missing or invalid.
1021	dial error	The primary and secondary (if specified) connections were dialed but the CCD could not connect the remote server.
1022	no dial tone	The phone line may be disconnected
1023	line busy	The dialed phone number is busy.
1024	carrier lost	The modem carrier were lost when the CCD was waiting for a message
1025	no carrier	The CCD could not negotiate a compatible protocol with the remote modem. Make sure you are using the correct connection speed, do you have to dial a 9 before the phone number? If Datapac is being used try using a different phone number in your area or a different speed .
1026	no answer	The dialed phone number is not answering
1027	datapac cleared	The datapac connection was cleared. The network could be busy, try later.
1028	script unsuccessful	The user connection script was unsuccessful. This does not mean that the script is wrong, this could be generated by a temporary network unavailability. See the log file for more details.
1031	Protocol error	A CDAnet version 2.0 or 4.0 protocol error occurred. This could be either a timeout or retry condition. (have been replaced by the appropriate status code 1041-1044
1032	Decompression error	There was not enough room for decompressing the message. The cause could be a corrupt response from the remote server or switch.
1033	err read input	Error reading the dental claim from (input file)
1034	Request invalid	The dental claim is not a valid CDAnet request. It contains control characters, the length field A07 is not properly set , a V2 transaction contains accented characters and the v2ConvertFrench is set to FALSE, a V2 file contains accented characters in an unsupported Code Page (e.g. other that Code Page 850)

		See the ccd.log file for a more detailed explanation of what is causing the rejection.
1041	ENQ timeout	The connection script was successful, but the server is not sending the prompt (no ENQ received) to start the transaction.
1042	Server timeout	The claim was sent but no response came back from the CDAnet network ‘s server.
1043	Invalid characters	No response from server: The transaction was sent to the CDAnet server, but other characters (garbage) were received instead of start of response.(STX)
1044	Bad LRC	after retries, the LRC in server’s response was still bad. (this could be caused by a bad telephone line)
1045	Server disconnect	The server sent an EOT to abruptly terminate the transaction.
1051	communication open error	The communication port could not be opened. For a serial com port: The port is not configured properly or it is already in use by another program. For a TAPI modem this could mean a problem with Modem installation or connection For a TCP/IP connection this could mean that there is no Internet connection, or the CDAnet Server or Switch cannot be reached for some other reason.
1052	com param error	Error setting the serial port parameters.
1053	com write error	Error during a write to serial port or modem
1054	com read error	Error reading from the serial port or modem

Error: Message 2011 Multiple CCD running

The CCD can now detect a problematic situation in the dental office where several CCD programs were started in the same directory.

For example:

- Multiple desktops in the same clinic start the CCD in the same shared directory;
- Two user accounts in two different sessions on the same computer starting up the CCD in the same directory.

What is the problem?

Having multiple CCD in the same directory results in sending multiple duplicate transaction to the CDAnet server. Indeed each CCD sees the transaction and sends it to the serrver, which in turn transfers it to the carrier who receives multiple identical claims.

Detection:

When the CCD detects this condition, it displays the following message.

The message gives:

- The directory of the CCD (in the example above "W: \ Program files \ ccd \ test")
- The name of the computer from which the CCD was first started (in the example above "FAWKS02")

What to do?

The software specialist or the person who configured the software to start the CCD, should correct the situation by using one of the following solution:

- Start a single CCD on the server in a shared working directory,
- Or, start a CCD at each workstation in a local directory.

The CCD detects the condition when it cannot open the file **activity.dat** exclusively in its working directory, so this message can also appear if this file is locked by another program (for example, a text editor) or if the user does not have the privilege to write to the file.

. CCD file description

The following list gives the name and description of files in the CCD directory:

File Name	Description
ccdupdate.dat and ccdupdate.exe	Programs and data for CCD automatic update. The program can update the **CCD32.exe, ccdredir.dat and ccdsup.ini** the old versions of this files are copied in $backup directory
ccdupdate.lck	Lock file reminder of the date of the last CCD update. At each startup the CCD checks for the file date and requests program update verification, if the file is older than 24 hours.
Ccdsup.ini	The supplemental initialization file contains additional configuration for supplemental CDAnet networks, for example Net+ This file has a version number that appears in the CCD/About Window. This file is updated during the execution of CCDupdate, so if it is deleted or modified, it will appear again the next time the CCD updates itself
ccdredir.dat	The forwarding table allows for redirection of some transactions to new Networks without reconfiguring the dental software. This file has a version number that appears in the CCD/About Window. This file is updated during the execution of CCDupdate, so if it is deleted or modified, it will appear again the next time the CCD updates itself
ccdstat.dat	Status file created by the CCD, contains the office number (CDAnet Field B02) used in the last transaction.
Ccdinstall.dat	CCDupdate creates this file. It gives the text status of the last update process so the CCD can display it on its log.
CCD Update Log.txt	CCDupdate creates this file. It is the log of the last update process.
Ccd32.id	CCDupdate creates this file. It contains a unique identification number for this CCD installation. If the file is deleted, it will be recreated during the next CCD update process.
$backup	This directory contains the latest versions of the files that where updated and used by CCDupdate if there is need to restore the installation.
_download	This directory contains the files downloaded by CCDupdate during the update process.

. Trouble shooting

In case of trouble you should examine the error code returned in the output file. You should also examine the content of the log file created by the CCD.

The log file

The log file (by default ccd.log) contains informational or error messages generated during transaction processing.
In order to use the log file you must enable message logging (see general parameters **logFile** and **enableLogging**).
You should also enable detailed message logging by doing one of the following:

- Using the **-v command line switch.**
- Selecting the **Help/detailed messages menu** in windows version.
- Setting the parameter **detailedMessages = TRUE** in the ini file.

Unless you set the parameter **logfileAppend** to true, the log file is erased at startup so make sure you save this file if you need to keep it for reference.

Apendix A Script syntax

The script language is used for direct or dialup connections that require special handling after the serial port and the modem is connected.
Script instructions are executed in sequence until the end of script is reached, or a 'return' statement is executed.
If you return a non-zero value, the CCD will consider the script as unsuccessful and will terminate the call.
If the end of the script is reached before a return statement is executed, the script is also considered as unsuccessful.

- The instructions can be specified in either lower or uppercase;
- your can add CR or indent the instructions anywhere except inside a string or numeric constant;
- several instructions can be grouped to form a bloc by enclosing the instruction between curly braces "{" and "}";
- you can insert control characters into a string by using the ^A to ^Z notation, for ascii characters 0 dec to 26 dec. (example "123^M")

Script syntax summary

Syntax	Example	Remark
; comment	`;this text ignored` `;****`	Everything typed after the ';' is ignored by the CCD.
Transmit [address] "<string>"	`transmit "TEST^M"`	this example transmits the string "TEST" followed by a carriage return character (Ctrl-M)
	`transmit address "^M"`	This example transmits the content of the field 'address' followed by a carriage return character.
Wait <n> for "<string>" then <statement_or_bloc>	`wait 3 for "DATAPAC:" then return 0`	Wait for the string "DATAPAC:" for a maximum of 3 seconds. If the string is received, the specified instruction or bloc of instruction is executed. In this case "return 0" Otherwise the next instruction is executed. If the end of script is reached, the script returns with a failure code.

	`Wait 3 for "DATA" then`
	`  wait 2 for "PAC" then`
	`  wait 2 for ":" then`
	`  return 0`
		this is almost equivalent to the last example

retry n <statement or bloc>	`retry 3 transmit "."`	This example sends 3 "." characters to the serial port

	`retry 4 {`
	`  wait 3 for "username:"`
	`  then{`
	`  transmit "NAM^M"`
	`  return 0`
	`  }`
	`  transmit "^M"`
	`}`
		retry 4 times:
		if the "username:" string is received in 3 seconds, send the string "NAM" and exist script (return 0 = successful)
		Otherwise transmit a CR and repeat the wait.
		After 4 retries, the script is terminated and is considered unsuccessful.

Return <n>	`return 12  ; unsuccessful`	Terminates the script and returns a status code.
Delay <n>	`delay 5`	Introduce a delay of 5 seconds in the execution of the sript.

Error checking

The script syntax checking is performed once during the launch of the CCD. An error message will give more detail about the syntax error, along with an error line number. Note that the actual error could be located on a previous line(s).