# ITRANS
# Communication Agent

## Installation and programming guide

Version : 2.0
Date : 2005. 01.07

| Revisions | | | |
|---|---|---|---|
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | 2002/06/27 | Paul Knapp | Original version |
| 1.1 | 2004.02.03 | Paul Knapp | Revisions to incorporate iCAControl versus iCA application |
| 1.2 | 2004.02.09 | Paul Knapp | Added IniFile property |
| 2.0 | 2005.01.07 | Dave Remmer | Added changes for HL7 Enabled Version |
| 2.0 | 2005.04.13 | Samir Georges | Added iCA Status Codes (1026 & 1045) |
| 2.0 | 2005.09.22 | Samir Georges | Added iCA Status Code (1025) |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Contents**

 

# Introduction

The iTRANS communication protocol (TCP/IP with SSL 3.0 or TLS 1.0) and message exchange standard (ebXML) has been designed to present a simple and unified way of transmitting messages between providers and payors.

The iTRANS Communication Agent or "iCA" is the client-side implementation of the protocol on the Windows platform.

This document is primarily intended to help software vendors to install and configure the iCA. The document also explains the mechanisms through which the dental office software should interact with the iCA.

## Key features

- The iCA implements TCP/IP, SSL or TLS and ebXML and can transmit CDAnet version 2.4, 3.0 and ACDQ-CDAnet version 4.0 transactions to the iTRANS host for forwarding to payors;
- The iCA application 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 iCA can be called through a COM+ control that allows direct programmatic interfaces for communication without needing to go through the directory and file scheme used by the application.
- The iCA can wrap the Common Communications Driver (CCD) application allowing one consistent interface for both asynchronous communications and TCP/IP communications.
- Finally, the ICA includes an extended log and trace file for developing and debugging the interface with your software.

## System Requirements

- Microsoft Windows: Win95/Win98/NT 4.0/2000/ME/XP
- 8 Mb of disk space
- Either a direct IP connection to the Internet (Cable Modem, ISDN, ADSL) or a dialup connection to an ISP configured to dial on demand

# 1. Getting Started

### Before installing the iCA

Make sure the computer is able to connect to the Internet either through an 'always on' (cable, ISDN, ADSL) or will connect on-demand through a dialup, modem, link to an ISP.

Make sure the iTRANS X.509 certificate has been installed on the local computer, the one which will run the iCA software. The certificate should be viewable from your Internet browser and appear in the MY certificate store for the CURRENTUSER.

If there are multiple providers in the office then a separate certificate is required to be installed for each provider.

### Installing the ICA

Decompress the iCA.zip archive file to the C:\iCA directory or use the existing C:\CCD directory (the INI settings are set to the C:\iCA directory by default). Insure the decompression program copies the software and support files. As well, many of the included controls need to be registered on the target operating system. See the attached ReadMe.txt file for more details.

Then add a shortcut to the iCA program to the STARTUP folder.

The Practice Management Software application should take care of this installation step in it's own Setup Application or process.

### Running the ICA

The iCA can be run from the command prompt. iCA

example:

C:\iCA\iCA.exe

If you create a shortcut to start the iCA, it is recommended you make the C:\ICA directory the default.

**Only a single instance of the program can run** at a given time, so restarting the iCA will not start a new instance. If it is required to have multiple instances of the iCA, for example to support multiple pools of users, you will have to start the second instance of the iCA from another directory (example C:\ICA2)

# 2. Configuration

On all platforms, the configuration of the iCA is stored in an ASCII file (by default iCA.INI ) which is a conventional Windows INI file. This file can be edited by using any text editor.

### ICA.INI format and syntax

The ini file is composed of three sections:

```
❶
[iCA]
param = value
  ... ;  general iCA parameters

❷
[Service]
param = value
  ... ;  service specific parameters

❷
[Carrier]
param = value
  ... ;  carrier specific parameters
```

| | |
|---|---|
| ❶ The general parameter section | This section contains the parameters that are common to all services that can be accessed through the ICA. For example which Service Profile to use. |
| ❷ The service profile section | This section contains one or more service profiles which contain the service specific parameters. For example the location that certificates are stored in. |
| ❷ The carrier profile section | This section contains one or more carrier profiles which contain the carrier specific parameters. For example the service this carrier uses, or whether to tunnel through the CCD application, or the carrier's IP destination. |

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

### General parameters section

**TranType= <Type of Transactions>**

The type of transactions that will be sent through the iCA. Allowable values: **CDANet, HL7**. For example:

```
TranType = CDANet
```

**CertTypeOID = <OID of Certificate Type>**

This contains the OID of the type of certificate the iCA will use to identify the sender. This value will normally be provided by the certificate issuer and will be the same for all the certificates issued to a type of provider and will not change.

**iCADirectory = <directory iCA will use as it's root directory>**

The directory that the iCA will use to store files, as well as reading temporary configuration files and CCD input and output.

**Debug= YES | NO**

Setting this value to YES will provide additional debugging information as well as fire Debug events from the COM+ control.

**ShareLineWithCCD = YES | NO**

This value determines whether the software will be sharing a telephone line with the CCD application. If so, the software will not "stay alive" after transmitting transactions.

**ControlCCD = YES | NO**

Setting this value to YES will have the iCA control the CCD application and transmit claims asynchronously through this application.

**TimeoutSeconds = <number of seconds>**

The number of seconds the iCA will wait before returning an error if no acknowledgement has been received.

**TransactionTimeoutSeconds = <number of seconds>**

The number of seconds between receiving an acknowledgement and receiving the response transaction from the host before returning an error.

**StayAliveSeconds = <number of seconds>**

The number of seconds the connection to the host will be left open while waiting for another transaction to be sent. After this number of seconds the connection will be closed.

**CCDTimeoutSeconds = <number of seconds>**

The number of seconds the iCA will wait for the CCD to transmit a transaction and receive a response from the host.

**Ver2OutstandingCarrierID = <carrier ID of the carrier to send Version 2 Outstanding calls to>**

CDA Version 2 Outstanding Transactions do not indicate which carrier / network they are for. This setting allows the iCA to select a carrier to send these transactions to.

**Log = YES | NO**

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

**AppendLog = YES | NO**

This is an optional parameter, if true, the ICA 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.

## Service profile parameters section

The iCA can be configured to connect to a number of services, with their service profiles stored in the INI file. This section of the INI file is where all parameters regarding each supported service are defined.

[ServiceName]

CertStore=MY

CertLocation=CurrentUser

The following sections define the service parameters.

**CertStore = MY | ROOT**

The 'CertStore' member identifies which certificate storage location to search for X.509 certificates for secure communications.

**CertLocation = CurrentUser | LocalMachine**

The 'CertLocation' member identifies which certificate storage area to search for X.509 certificates for secure communications.

## Carrier profile parameters section

The following parameters are set for each carrier that can be communicated to through the iCA. These settings may be for the carrier themselves, or a host that is providing communication services on behalf of the carrier.

The carrier's OID also needs to be listed in the [OIDs] section, in order to cross-reference it to this section.

**CarrierNetwork= <directory for storing files for CCD>**

The directory that the iCA will place files in when communicating through the CCD. If the carrier does not go through the CCD then the name of the service should be placed here.

**CarrierOID = <OID for Carrier>**

This is the OID that is set in the receiver field of the message when the transaction is not being sent through the CCD.

**UseCCD = YES | NO**

Indicates whether the CCD should be used to transmit the transaction asynchronously. If NO, then the service described in the CarrierNetwork field will be used.

**PrimaryIP = <IP address of the carrier or network>**

IP address the iCA will communicate to first if the transaction is not a CCD transaction.

**PrimaryPort = <Port number the carrier or network is listening on>**

The port number the iCA will communicate through to communicate with the Primary IP address.

**SecondaryIP = <IP address of the carrier or network>**

IP address the iCA will communicate to if the Primary location did not respond and the transaction is not a CCD transaction.

**SecondaryPort = < Port number the carrier or network is listening on >**

The port number the iCA will communicate through to communicate with the Secondary IP address.

**IssuedTo = <name of who the host certificate was issue to>**

This is the name of the issue to portion of the host's digital certificate to be verified against the host's certificate when connecting.

**IssuedBy = <name of who the host certificate was issued by>**

This is the name of the issuer of the host's digital certificate to be verified against the host's certificate when connecting.

# 3. Interfacing with the iCA

## Introduction

There are two methods of interfacing with the iCA: through the iCA Application which reads in files, transmits them through the iCA Control, and writes out files an application can then read (this method is how the Common Communications Driver works); or through the iCA Control which provides COM+ methods and properties to allow programmatic control.

## iCA Application Interface

The iCA software connects to only one remote system as specified in the Profile parameter. For each named profile there must be a subdirectory in the iCA main directory having the name of that profile.

The office software creates the input file (eg. dental claim) in the desired profile subdirectory.

The name of this file must be INPUT. The extension of this file is a 3 character (LUN) Logical Unit Number. The iCA will create the response file in the same subdirectory, the name of this file is OUTPUT. The extension is the same as the input file.

For example, in order to send a transaction to iTRANS:

1. Configure and run the iCA, make sure it runs in the \ICA directory
2. Delete any older output file that may remain \ICA\OUTPUT.000
3. Create a file named \ICA\INPUT.000
4. Wait for the response file \ICA\OUTPUT.000
5. For sending another transaction go to step 2
6. Stop the iCA and examine the content of \ICA\iCA.LOG see trouble shooting section.

Note 1: The input file is opened exclusively only during the time required to ready the contents.

It should also be noted that the iCA Application actually wraps the iCA Control, and so the transmission of transactions always go through the iCA Control to the host.

## iCA Control Interface

The iCA Control implements a COM+ control for communicating to the carrier or network. It allows the setting of properties to override the settings in the INI file. The **SendTransaction** method will begin the communication to the host, and the **ReceivedResponse** event will contain the results from the host. If an error occurs the **Error** event will be received.

Generally, a calling application will do the following to use the iCA Control:

1. Instantiate an iCA Control.
2. Override any of the Properties that have been loaded by the Control from the iCA.ini file.
3. Call the **SendTransaction** method with the string containing the transaction to send to the host.
4. Wait for the **ReceivedResponse** event containing the string with the transaction results.

5. Catch any **Error** events that might have occurred. If an error event occurs, no ReceivedResponse will occur.

**Methods**

**SendTransaction(BSTR szTransaction)**

This method contains the transaction text to be sent to the carrier or network. See the section below for a description of its format.

**Properties**

**IniFile : BSTR**

This property is the one property that (for obvious reasons) is not loaded from the INI file. This allows the created control to have a specific INI file set for it; after setting this property the INI file will be reloaded so any properties that have been set through the COM+ interface will be reset with their corresponding value from the INI file.

**TranType : BSTR**

This property sets the Transaction Type. See the description in the General Parameters Section of the Configuration chapter.

**PrimaryIP : BSTR**

This property sets the Primary IP address. See the description in the Carrier Parameters Section of the Configuration chapter.

**PrimaryPort : BSTR**

This property sets the Port for TCP/IP communications. See the description in the Carrier Parameters Section of the Configuration chapter.

**SecondaryIP : BSTR**

This property sets the Secondary IP address. See the description in the Carrier Parameters Section of the Configuration chapter.

**SecondaryPort : BSTR**

This property sets the Port for TCP/IP communications. See the description in the Carrier Parameters Section of the Configuration chapter.

**iCADirectory : BSTR**

This property sets which directory will be used by the iCA for basic file routines. See the description in the General Parameters Section of the Configuration chapter.

**Debug : BOOL**

This property sets whether debugging support is turned on or not. See the description in the General Parameters Section of the Configuration chapter.

**ShareLineWithCCD : BOOL**

This property indicates whether the iCA is sharing a telephone line with the CCD or not. See the description in the General Parameters Section of the Configuration chapter.

**ControlCCD : BOOL**

This property indicates whether the CCD application should be controlled by the iCA or not. See the description in the General Parameters Section of the Configuration chapter.

**Log : BOOL**

This property turns on or off the logging functionality of the iCA. See the description in the General Parameters Section of the Configuration chapter.

**AppendLog : BOOL**

This property indicates if the log should be reset when a new transaction is started (NO) or if the log file should be appended to instead (YES). See the description in the General Parameters Section of the Configuration chapter.

**TimeoutSeconds : LONG**

The number of seconds before an error is issued if an acknowledgement is not received. See the description in the General Parameters Section of the Configuration chapter.

**TransactionTimeoutSeconds : LONG**

The number of seconds between an acknowledgement being received and a transaction response being received without an error being issued. See the description in the General Parameters Section of the Configuration chapter.

**StayAlive : LONG**

The number of seconds the iCA will keep a connection open with a host waiting for an additional transaction to that host. See the description in the General Parameters Section of the Configuration chapter.

**CCDTimeoutSeconds : LONG**

The number of seconds the iCA will wait for the CCD to complete a transaction. See the description in the General Parameters Section of the Configuration chapter.

**Events**

**ReceivedResponse(BSTR szTransaction)**

An event containing a string with the transaction response data.

**Error(long nErrorCode)**

An event with a code that indicates an error occurred while processing the transaction. Please note these errors are “communication errors”, not transaction errors. The list of errors that might be returned are described below.

**Debug(BSTR szDebugString)**

An event containing a string with the debugging information. Please note this event will only be generated if the Debug property is set to YES.

## PC network setup

The PC network setup is similar to a single workstation setup. One PC acts as a iCA server, and shares its ICA directory with the other PC's in the network.

Each PC creates an input file in this shared directory, then waits for the result in the same directory.

In order to prevent one PC from overwriting 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:\ICA\ INPUT.001

PC 002 creates N:\ICA\ INPUT.002

The ICA handles the input files one at a time. The result for PC001 is created in N:\ICA\\OUTPUT.001 and so on.

## Input file format

The input file is composed of a single message, eg. A CDAnet dental claim, on a single line starting at the first character of the line, with no embedded new-line characters.

Only the first line (up to the end of line) is read by the iCA. 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
```

## Output file format

The output file is composed of several comma-separated fields followed by a single response message, eg. A 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 payor's response, (only when the transaction could be sent to the payor).

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 payor’s response to your message.

## 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 iCA. (see Output file format for more details)

**Table 2- iCA 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 | Connection timeout | The iCA was not able to establish a connection with the remote site during the specified time frame. If using a dial-up connection increase the timeout period. |
| 1011 | Remote certificate invalid | An invalid certificate has been supplied by the remote site, or the certificate issuer is otherwise unable to validate. Check the address for the remote site then contact technical support if unresolved. |
| 1012 | Local certificate invalid | An invalid certificate has been supplied to the remote site, or the certificate issuer is otherwise unable to validate. Check the certificate then contact technical support if unresolved. |
| 1013 | Certificate not found | The connection requires a valid X.509 certificate for each sending provider and office combination. The software could not locate a valid certificate. |
| 1021 | Connection refused | The iCA could not connect to the remote server, check the address. |
| 1024 | Circuit Reset | The connection was broken or terminated by the remote site. |
| 1025 | Host Has Refused Connection | A 1025 iCA error code occurs under two different conditions:<br><br>First Condition:<br>You've attempted to connect to the wrong site. By wrong site we mean that you might be sending to an Internet address that isn't an ITRANS server (i.e. no server listening on port 9650). This can be caused if your DNS server has the wrong IP information for the site domain name, and/or there's a firewall blocking port 9650.<br><br>Second Condition:<br>Invalid authentication information in the iCA.INI for the specified carrier(s). The following fields might have the wrong information: IssuedTo, and IssuedBy.<br><br>The proper information for these fields is:<br>IssuedTo=CSI ITRANS Site<br>IssuedBy=CSI CA<br><br>The 1025 could be also caused by a combination of both. |
| 1026 | Invalid Carrier | Incorrect or Invalid Carrier ID. |
| 1034 | Request invalid | The dental claim is not a valid CDAnet request. See the log for details as to whether the problem is transaction length, version, provider, office or payor related |
| 1041 | ACK timeout | The message was sent to the remote site but not acknowledged within the specified time frame. |
| 1042 | Server timeout | The message was sent and acknowledged but no message response came back from the host's server. Consider increasing the specified time frame. |

| | | |
|---|---|---|
| 1045 | Server Timeout | The iCA has received a message, but is unable to send the message. This is often caused by missing ".tmp" files in the iCA directory specified in the iCA.ini. |

# 4. 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 iCA. It is also generally helpful to turn on debugging to get more detailed descriptions of what is transpiring within the iCA.

### The log file

The log file (named iCA.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 **Log**).

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