

CCDWS

Common Communication Driver for Web Services Technical Reference and Operating Manual

Table of Contents

1. Introduction	1
2. Run-Time Environment	1
2.1. Operating Systems	1
2.2. Required Run-Time Libraries	1
2.3. Location of Run-Time Files	1
2.4. Location of Certificates	2
3. File Communications	2
3.1. File Location	2
3.2. Input File Generation	2
3.3. Input File Format	2
3.4. Output File Format	3
4. REST Communications	3
4.1. REST Schema	3
4.2. Sample Request Message Sent to CCDWS	4
4.3. Sample Response Message Received from CCDWS	4
4.4. REST EndPoint	4
4.5. Transfer Character Set	4
4.6. REST Listener Certificate	5
5. Web Service (SOAP) Communications	5
5.1. WSDL	5
5.2. Sample Request Message Sent to Remote Server	6
5.3. Sample Response Message Received from Remote Server	6
6. Error Handling	7
7. Configuration	7
7.1. Configuration File Format	7
7.2. Configuration Reference	8
7.3. Sample Configuration File	19
8. Automated Installation Instructions	20
8.1. Before Installation	20
8.2. Installing CDA Digital IDs to the Certificate Store	21
8.3. Installing Vendor Certificates	21
8.4. Removing old Installations of CCDWS	21
8.5. Running the Setup Program	21
8.6. Accepting the EULA and Reading Installation Notes	22
8.7. Setup Type	22
8.8. Destination Folder	22
8.9. Service Account	22
8.10. Actions Performed on Installation	22
8.11. Configuration File Review and Edit	22
8.12. Completion and Starting the CCDWS Service	23
8.13. Additional Tasks Upon Successful Installation	23
9. Manual Installation Instructions	23
9.1. Before Installation	23
9.2. Extracting Files	23

9.3.	Installing CDA Digital IDs to the Certificate Store or Keystore	24
9.4.	Installing Vendor Certificates.....	24
9.5.	Updating the CCDWS Configuration File.....	24
9.6.	Updating the Existing CCD Configuration File.....	24
9.7.	Confirming CCDWS Operation.....	24
9.8.	Installing the CCDWS Service	25
10.	Controlling the CCDWS Service	25
10.1.	GUI Control	25
10.2.	Command Line Control	26
11.	Executing CCDWS from the Command Line.....	26
12.	Log Files	26
12.1.	Log File Format	26
12.2.	Configuring Log Filenames and Levels	27
12.3.	Log File Rotation.....	27
12.4.	Sample Log File with Notes	28
13.	Status Codes.....	29
14.	Internal Error Codes.....	30
15.	Troubleshooting	33
16.	OpenSSL License.....	35
17.	gSOAP License	37
18.	CCDWS End User Licensing Agreement (EULA).....	37

Revision History

Date	Revision	CCDWS Version	Changes
2011-10-25	1.0	0.9.0 (beta)	❑ Initial revision
2011-11-03	1.1	0.9.0 (beta)	❑ Removed OpenSSL run-time library requirement
2011-11-07	1.2	0.9.1 (beta)	❑ Changed SignMessage option ❑ Added <i>Related Documents</i> section
2011-11-10	1.3	0.9.2 (beta)	❑ Added <i>Prefix</i> and <i>DataType</i> configuration options
2011-11-16	1.4	0.9.3 (beta)	❑ Removed <cr> from output file format specification
2011-12-12	1.5	0.9.4 (beta)	❑ Changed <i>SignMessage</i> configuration option to <i>SignMessageParts</i> ❑ Added <i>EncryptMessageParts</i> , <i>EncryptMessageBody</i> , and <i>EncryptCertificateFile</i> configuration options
2012-01-11	1.6	0.9.4 (beta)	❑ Added <i>Automated Installation Instructions</i> section ❑ Added other related documents ❑ Updated sample request and response messages ❑ Changed <i>DataType</i> configuration option to <i>PayloadType</i>
2012-01-31	1.7	0.9.5 (beta)	❑ Added <i>DetectParseErrors</i> configuration option
2012-05-30	1.8	0.9.6 (beta)	❑ Changed <i>MaxRetries</i> configuration option description to add conditions under which retries will be attempted. ❑ Changed default <i>ReadWriteTimeout</i> to 60 seconds.
2012-05-30	1.9	0.9.6 (beta)	❑ Fixed page footer text.
2012-08-06	1.10	0.9.6 (beta)	❑ Added missing <i>RootDir</i> configuration option. ❑ Changed <i>ProxyAddress</i> configuration option to <i>ProxyServer</i> ❑ Set <i>ProxyServer</i> and <i>ProxyPort</i> configuration option section to <i>Main</i> rather than <i>Destination X</i> .
2014-07-29	1.20	1.0.1	❑ Added description of CDA specified certificates in operating system's certificate store ❑ Added configuration options <i>ProviderOidPrefix</i> , <i>OfficeOidPrefix</i> , <i>CaCertificateName</i> , <i>RootCaCertificateName</i> , <i>CrlAltDistributionPoint</i> , <i>CrlDistributionPoint</i> , and <i>CrlRefreshDelay</i> ❑ Updated descriptions for <i>CaCertificateFile</i> , <i>CertificateFile</i> ❑ Added and updated sections for automated and manual installation to reflect the use of CDAnet certificates. ❑ Added <i>Troubleshooting</i> section
2014-10-10	1.21	1.0.1	❑ Added troubleshooting entry where certificate is in the CRL.
2014-10-22	1.22	1.0.1	❑ Added additional information in installation sections detailing location of installation files, and describing conditions for installing pre-1.0 functionality.
2014-11-28	1.23	1.0.1	❑ Added troubleshooting item regarding endpoints or certificates not matching the appropriate operating system. ❑ Added troubleshooting item regarding no valid certificate being found (either expired or missing). ❑ Changed error code for key marked as not exportable in troubleshooting guide to 1013. ❑ Changed error code for certificate not found to 1013

Date	Revision	CCDWS Version	Changes
			<ul style="list-style-type: none"> Updated SSLVerify configuration option description.
2014-12-01	1.24	1.0.1	<ul style="list-style-type: none"> Added configuration file sample Additional references to TELUS network
2014-12-03	1.25	1.0.2	<ul style="list-style-type: none"> Added Foreground configuration option.
2015-05-12	2.1	1.0.3	<ul style="list-style-type: none"> Added information for installing and operating within MacOS.
2015-11-12	2.2	1.0.5	<ul style="list-style-type: none"> Added Linux in addition to MacOS information. Changed copyright notice and other references from HIEC to CLHIA
2016-11-11	2.3	1.0.5	<ul style="list-style-type: none"> Additional minor changes from HIEC to CLHIA
2017-08-18	2.4	1.0.6	<ul style="list-style-type: none"> Added CertificateFileProviderPrefixes, CertificateFileEndPoint, and CertificateFileAction configuration properties Added EULA to document Modified installation sections to provide more detail on installing vendor certificates. Created additional troubleshooting item to handle case where CertificateFileProviderPrefixes is not defined.
2017-11-21	2.5	1.0.6	<ul style="list-style-type: none"> Added ProxyUserId and ProxyPassword configuration options in the configuration reference. Minor change to troubleshooting section for Proxy Authentication error.
2018-07-18	2.6	1.0.7	<ul style="list-style-type: none"> Added InstalledComputerName configuration option
2021-03-22	3.0	2.0.0	<ul style="list-style-type: none"> Added information related to REST client side communications, including description, schema, config reference.
2021-08-05	3.1	2.0.0	<ul style="list-style-type: none"> Additional REST config options CertificateCN and CertificateIssuer
2021-09-27	3.2	2.0.0	<ul style="list-style-type: none"> Added CertificateExportRetries and CertificateExportRetrySleep configuration options
2021-11-17	3.3	2.0.1	<ul style="list-style-type: none"> Removed CrlDistributionPoint and CrlAltDistributionPoint configuration options as they are retrieved directly from certificates. Fix page numbering to start at 1
2022-06-14	3.4	2.0.2	<ul style="list-style-type: none"> Fixed REST sample input added [Listener] as possible section for various log related configuration options Updated sample config file
2022-07-11	3.5	2.0.2	<ul style="list-style-type: none"> Update REST Listener section with corrections and improved detail.
2023-08-08	3.6	2.0.3	<ul style="list-style-type: none"> Replace OS X with MacOS Remove assumption regarding Windows in manual installation instructions. Update locations of files to include those for MacOS and Linux

Deleted: OS X

Deleted: OS X

1. Introduction

This document provides technical and operational information for the Common Communications Driver for Web Services (CCDWS), sponsored by the Canadian Life and Health Insurance Association Inc. (CLHIA), and designed by Harder Software. CCDWS is designed to act as a communications interface between dental offices and healthcare insurers.

CCDWS is designed to run as a service application, supporting two forms of communication between clients applications: listening for input REST connections and also polling for input request files. Upon acceptance of input request it will perform all tasks necessary to submit the claim and receive the adjudication response or acknowledgement in real time. With its file communications it will operate in a parallel with the original CCD application, so that both carriers that accept dialup claims and those that accept web service claims can be supported simultaneously without modification to the dental software. As CCD does not support REST communications, this form of communication is not backward compatible.

2. Run-Time Environment

2.1. Operating Systems

CCDWS is currently compiled to execute in recent versions of 32 or 64-bit Windows, 64-bit Mac MacOS, and 64-bit Linux.

Deleted: OS X

2.2. Required Run-Time Libraries

All object libraries for the CCDWS executable file are statically linked, so that no additional run-time libraries beyond those provided by the operating system are required.

2.3. Location of Run-Time Files

CCDWS files can be installed in any local filesystem location, and the location of files can be controlled during the automated Windows installation process. Additionally, the configuration file location can be specified as a command-line option. However, the default values will allow CCDWS to run seamlessly alongside existing Windows CCD installations.

The file locations are as follows:

<root>/<ccdws_exe>: the CCDWS executable for Windows and MacOS
<root>/ccdws.ini: the CCDWS configuration file for Windows
<root>/ccdws.conf: the CCDWS configuration file for MacOS
/usr/local/etc/ccdws.conf: the CCDWS configuration file for Linux
<root>/ccdws.log: the CCDWS log file for Windows and MacOS
/var/log/ccd/ccdws.log: the CCDWS log file for Linux
<root>/<dest>/input.<ext>: input file containing the request payload (if using file communications)
<root>/<dest>/output.<ext>: structured output file corresponding to the input file (if using file communications)

Where:

`<root>` is the root directory containing the executable file and others, as well as the destination directories. For Windows and MacOS, this defaults to the current working directory. For Linux, this defaults to `/var/spool/ccd`

`<ccdws_exe>` is the CCDWS executable. In Windows this is `ccdws.exe`. In Linux and MacOS, this is `ccdws`.

`<dest>` is the destination directory, referenced by the `[Destination <dest>]` directory in `ccdws.ini/ccdws.conf`. The value of `<dest>` will often be a carrier abbreviation. For instance ABC = Alberta Blue Cross. The carrier TELUS has many alternative values such as NDC, SHNS, etc.

`<ext>` is the input/output file extension given by the dental software.

Deleted: OS X

2.4. Location of Certificates

For Windows, CDA provides an installation application that should be used to install both the CA (certificate authority) and provider Digital ID certificates into the operating system's certificate store. It is important that these certificates be installed correctly prior to using CCDWS.

If certificates are installed in the current user's environment it is important that CCDWS be installed such that it runs under the current user's account, regardless of whether it runs as a simple executable or as a service. If the certificates cannot be found, then communications will fail.

For Linux and MacOS, CDA provides a Java application to create or update both a `cacerts` keystore file containing CA certificates, and a `providerkeys` keystore file containing provider certificates and keys. These are created in the current working directory of the Java application. They can be copied to the application's root directory, or any other location that is specified in the configuration file.

Deleted: OS X

3. File Communications

The CCDWS application can communicate with the dental office software using two methods: file based inputs and outputs, and a REST interface. File based inputs and outputs are consistent with the Windows CCD application, allowing coexistence of CCD with CCDWS, and seamless integration of CCDWS within an existing dental software installation.

3.1. File Location

The input files are obtained within the directory structure defined by the configuration option `RootDir`. In Windows the default is `c:\ccd`. Beneath that structure are destination directories. CCDWS will search for the input files based on sections within the configuration file. For instance, if a section `[Destination XYZ]` exists, then CCDWS will look for input files within `c:\ccd\XYZ`. It will do so for every destination section it finds.

3.2. Input File Generation

The CCDWS application expects that input files will be accessible only when they are generated in their entirety. This can be accomplished from the dental software by writing to a temporary file, then renaming the file to the correct naming convention as described below in the *Input File Format* section. CCDWS will also attempt to ensure it is not reading from a file being written to by first opening it in write mode, but this may not be a viable locking mechanism in all operating systems.

3.3. Input File Format

The input file is given the name `input.xxx` where `input` is the base part of the filename (i.e. 'input'). This can be altered with the configuration option `InputFileName`, but will not typically need to be other than the default. `xxx`

is a unique suffix within the input directory. Once an input has been processed, the first character of the file will be changed to an underscore ('_') so that it is not processed again.

The content of the file will be a valid CDAnet input, such as a claim request. CCDWS will accept CDAnet v2, v3, or v4 inputs so the dental office software is responsible for ensuring that the version is acceptable to the remote carrier. CCDWS will parse the input to ensure all fields are of appropriate types and mandatory fields are provided as per the standards, and to obtain the office sequence number used to populate the output file.

3.4. Output File Format

The output file will be generated for every input, unless the CCDWS is prevented from doing so due to file permissions or similar errors. Thus, even failures to successfully parse an input or failure to successfully connect to the remote carrier will result in an output.

The format of the output file will be *output.xxx* where *output* is the base part of the filename (i.e. 'output'). This can be altered with the configuration option *OutputFileName*, but will not typically need to be other than the default. *xxx* is the same extension as that of the input file. If the file already exists it will be overwritten.

The content of the file will be as follows: *<sequence>*,*<status>*,*<extension>*,[*<response>*]

Where:

<sequence> is the office sequence number extracted from the input file (for non-CDAnet, this is 0)

<status> is the numerical status of the transaction, where zero (0) is success.

<extension> is the file extension

<response> is the response payload to the input, if *<status>* is zero (0).

See the section entitled *Status Codes* for details on possible status code values and their meanings.

4. REST Communications

The second, and most recent, method of communication between dental software and the CCDWS is REST interface. To support this, the CCDWS accepts connections much like HTTP(S) calls, with claim details embedded in the message, and responds in a similar manner. The messages are structured JSON, and the payloads are the same as those sent and received by file I/O with the exception of UTF-8 encoding.

4.1. REST Schema

The REST schema is defined as follows:

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "type": "object",
  "title": "CCDWS REST Communications",
  "description": "Common format to handle dental claims",
  "examples": [
    {
      "input": "<CDAnet dental formatted string>"
      "destination": "<destination id>"
    },
    {
      "output": "<CDAnet dental formatted string>"
    }
  ],
  "definitions": {
    "common": {
      "type": "object",
      "properties": {
        "destination": {
          "type": "string",
```



```

        "description": "Contains the message destination identifier.",
        "examples": ["TELUSA", "TELUSB", "ABC"]
    },
    "input": {
        "type": "string"
        "description": "Contains the CDAnet request message",
        "examples": ["HD*..."]
    },
    "output": {
        "type": "string"
        "description": "Contains the CDAnet response message with CCD header.",
        "examples": ["1,0,,HD*..."]
    }
},
"additionalProperties": false
},
"request": {
    "allOf": [
        {"$ref": "#/definitions/common"},
        {"properties": {"output": false}},
        {"required": ["input", "destination"]}
    ]
},
"response": {
    "allOf": [
        {"$ref": "#/definitions/common"},
        {"properties": {"input": false, "destination": false}},
        {"required": ["output"]}
    ]
}
}
}

```

4.2. Sample Request Message Sent to CCDWS

```

{
  "input": "ABCCDANET4 0006010401000105DX1100632...",
  "destination": "ABC"
}

```

4.3. Sample Response Message Received from CCDWS

```

{
  "output": "601,0,,ABCCDANET4 0006010411000105001590133200411...",
}

```

4.4. REST EndPoint

The endpoint for sending messages will be based on the hostname that CCDWS is running on. If the sending application is on the same host, then this may be called "localhost." Otherwise it can be the locally defined hostname known on the network, for example "myhost." The default port is 9000 and default path is /dentalClaim. These can be changed using the Port and Path configuration options in the [Listener] section of the configuration file.

An example endpoint, if the host is called "myhost" would be:

```
https://myhost:9000/dentalClaim
```

When sending to the endpoint, use the POST verb.

4.5. Transfer Character Set

During REST communications, all messages translated to the UTF-8 character set. This means that special characters must first be translated from the default CP850 encoding that is specified for CDAnet data (and back to CP850 for responses). This may increase the size of the payload but the actual fields lengths are determined

based on CP850. Do not attempt to make an expanded UTF-8 field fit into the defined field size. For example, if a field size is defined as 10 bytes and the expanded UTF-8 field is 12 bytes, then the field should be transmitted as 12 bytes.

4.6. REST Listener Certificate

When installing provider certificates, a certificate with the common name 'CDA ITRANS Security' will be installed. CCDWS uses this to encrypt its communications.

The sending application should ignore any host validation errors for this certificate as it will not match that of the CCDWS host.

To disable encrypted communications between an application and CCDWS during development or troubleshooting, set the configuration option `SSLEnabled = 0` in the [Listener] section of the configuration file.

5. Web Service (SOAP) Communications

The CCDWS application is designed to support web service communications between itself and the network carriers. This means that the dental office must be able to access the internet.

Web services transmit XML encoded messages use the Simple Object Access Protocol (SOAP), typically transported using HTTP over TCP/IP in conjunction with other web standards, such as TLS/SSL secure communications. Additionally, web service extensions such as WS-Addressing and WS-Security are used to support secure delivery of messages. Response to the web services are returned similarly.

5.1. WSDL

The following WSDL defines the structure of the SOAP communications between the CCDWS and remote server:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="urn:hiec:v1" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:hiec:v1" name="HIECService" targetNamespace="urn:hiec:v1">
  <wsdl:types>
    <xs:schema version="1.0" targetNamespace="urn:hiec:v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
      <xs:complexType name="HIECTransaction">
        <xs:all>
          <xs:element name="Format" nillable="false" type="xs:string"/>
          <xs:element name="Value" nillable="false" type="xs:string"/>
        </xs:all>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="HIEC_Input">
    <wsdl:part name="request" type="tns:HIECTransaction"/>
  </wsdl:message>
  <wsdl:message name="HIEC_Output">
    <wsdl:part name="response" type="tns:HIECTransaction"/>
  </wsdl:message>
  <wsdl:portType name="HIECServicePort">
    <wsdl:operation name="HIECService">
      <wsdl:input message="HIEC_Input" xmlns:ns1="http://www.w3.org/2006/05/addressing/wsdl"
wsaw:Action="urn:hiec:v1:HIECService"/>
      <wsdl:output message="HIEC_Output" xmlns:ns1="http://www.w3.org/2006/05/addressing/wsdl"
wsaw:Action="urn:hiec:v1:HIECServiceResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="HIECServiceSoapHttp" type="HIECServicePort">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  </wsdl:binding>
</wsdl:definitions>
```

```

        <wsaw:UsingAddressing wsdl:required="true"/>
        <wsdl:operation name="HIECService">
            <soap:operation soapAction="urn:hiec:v1:HIECService"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="HIECService">
        <wsdl:port name="HIECService" binding="HIECServiceSoapHttp">
            <soap:address location="https://hiec.ab.bluecross.ca/P/HIECService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>

```

5.2. Sample Request Message Sent to Remote Server

The following message is generated by wrapping the CDAnet payload in a SOAP envelope, with a WS-Addressing extension. The actual payload is truncated with ellipses (...).

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa5:MessageID xmlns:wsa5="http://www.w3.org/2005/08/addressing">
      uuid:86e96a14-f36d-40e1-93f3-cf511f0a312c
    </wsa5:MessageID>
    <wsa5:To xmlns:wsa5="http://www.w3.org/2005/08/addressing" SOAP-ENV:mustUnderstand="1">
      https://vendor.claimstream.ca:3540/DWST/CDAnetService
    </wsa5:To>
    <wsa5:Action xmlns:wsa5="http://www.w3.org/2005/08/addressing" SOAP-ENV:mustUnderstand="1">
      urn:hiec:v1:HIECService</wsa5:Action>
    </wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ccd:request xmlns:ccd="urn:hiec:v1">
      <ccd:Format>
        CDANET4
      </ccd:Format>
      <ccd:Value>
        ABCCDANET4 0133200401000094EX5100500...
      </ccd:Value>
    </ccd:request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

5.3. Sample Response Message Received from Remote Server

The following sample message was received from the remote server. HTTP headers are not shown, and the message within the <ccd:CDAnet_RESP> tags is truncated with ellipses (...). The message is also formatted for easier reading, as it was received without line feeds and indents.

```

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing">
      www.w3.org/2005/08/addressing/anonymous
    </To>
    <Action xmlns="http://www.w3.org/2005/08/addressing">
      urn:cda-org:v1/CDAnetService/CDAnet_REQResponse
    </Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">
      uuid:WLS1:WseeJaxwsFileStore_auto_2:cf3d8cb7529a97cc:-5c264069:1333bb18cb8:-7ff2
    </MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">
      uuid:f6b3829a-9141-4090-9a4a-1e581ffec00a
    </RelatesTo>
  </S:Header>
  <S:Body>

```

```

<ccd:response xmlns:ccd="urn:hiec:v1">
  <ccd:Format>
    CDANET4
  </ccd:Format>
  <ccd:Value>
    ABCCDANET4 0133200411...
  </ccd:Value>
</ccd:response>
</S:Body>
</S:Envelope>

```

6. Error Handling

Errors encoded in the transaction responses are not detected as error conditions by CCDWS, and are sent back to clients applications with no modifications.

In the event of detected errors, CCDWS will write a non-zero status code back to the output file or attribute. To be compatible with CCD, these are limited to the subset of codes that are relevant to web services. As web service errors may be different than those supported by the CCD code set, a general error code will often be returned with details stored in the log file. For status codes and internal error codes, see the section entitled *Status Codes and Internal Error Codes*.

7. Configuration

While starting the CCDWS application, a command line option (`-c <configFile>`) may be provided to identify the location and name of the configuration file. In Windows, if combined with the installation parameter (`-i`) it will store this information in the registry for starting as a service. If run in command line mode, it will use that location upon start up. If no location is provided, it will look for the configuration file at `c:/ccd/ccdws.ini` in the Windows environment.

The CCDWS application must be restarted for any changes in the configuration file to take effect.

7.1. Configuration File Format

The format of the configuration file is similar to a Windows .ini file. There are section names and properties within those sections. Empty lines and lines beginning with ';' or '#' are ignored. However, there are differences from the standard .ini format. These can mainly be summed up as the ability for sections to inherit global properties, or properties from other sections:

Section and name values are insensitive, but the values may not be, depending on the context. For instance, usernames and passwords will typically be case sensitive, while filenames may not be in a Windows environment.

There is a global properties area at the top of the file. This acts as a default for properties read from any section, and is useful for settings that you may wish to use on a global level.

Each section may inherit properties from another section with an *InheritSection* property. This acts much like the global properties area, whereby values are read from that section. As an example, if we have two destination sections [Destination ABC] and [Destination NDC], they may both have the property *InheritSection=Destination Common*. The [Destination Common] section may contain the property *ConnectTimeout=10* meaning that the connection timeout for both of those destinations is set to 10 seconds. These inherited properties may be overridden in the calling section. For instance, a *ConnectTimeout* setting in [Destination ABC] or [Destination NDC] will override that in [Destination Common].

Note that the order of properties in a section is not important, and neither is the order of sections. The one exception is the global properties must appear before any section name.

As the configuration file may contain passwords, it is important that casual users should not have access to it.

Note that the existence of a [Listener] section will determine whether the new REST listener will be enabled. Regardless of this, the file based polling will always occur for backward compatibility.

7.2. Configuration Reference

Configuration properties and descriptions are shown below. See the section *Sample Configuration File* to view some common configuration properties in context.

The *Global* section is ultimately searched for by all properties as a default. This is not shown in the Sections listing for each property unless it is the only applicable section.

Action

Sections: [Destination X]

Default Value: urn:cda-org:v1:CDANet_REQ

Required: No

Description: The web service requires both an endpoint and an action. The default action need only be changed if the carrier defines a WSDL with different action value.

See Also: CertificateFileAction, Endpoint

Authentication

Sections: [Destination X]

Default Value: None

Required: No

Description: This defines the authentication scheme used by the remote server. Current valid values are 'UsernameTokenDigest', 'UsernameTokenText', and BinarySecurityToken. If no property is defined, no authentication scheme is included in the request. This will only be applicable if WS-Security is enabled.

CACertificateFile

Sections: [Destination X]

Default Value: None

Required: No

Description: Specifies the file name for the certificate authority (CA) certificate, if the SSL/TLS is used in the destination protocol (i.e. if the destination URL begins with https://). If an absolute path is not given, the file will be found relative to the destination directory. If this option is not provided, the operating system's certificate store (or Linux and [MacOS](#) CAKeystoreFile) will be searched using the values of CACertificateName and CARootCertificateName. The contents of CACertificateFile are expected to be in PEM format.

See Also: CertificateFile, CertificateKeyPassword, SSLVerify, CACertificateName, RootCACertificateName.

CACertificateName

Sections: [Destination X]

Default Value: CanDentAssocCA or CanDentAssocTestCA

Required: No

Description: Specifies the name of the certificate authority certificate used to generate the client certificate(s) stored in the certificate store. The default value depends on the certificate that may have been automatically retrieved from the certificate store based on the input message. This will be overridden if the value of CACertificateFile is provided. There may also be a root certificate authority

Deleted: OS X

certificate in the store that is used to generate this one. This value is ignored if *SSLVerify* is disabled.
See Also: RootCACertificateName, CACertificateFile, CertificateKeyPassword, CAKeystoreFile, KeystorePassword, SSLVerify

CAKeystoreFile

Sections: [Main]

Default Value: cacerts

Required: Yes, for Linux and **MacOS**

Description: For Linux and **MacOS** installations, specifies the name of the Java keystore file containing trusted certificate authority certificates. If no path is given, it is expected to reside in the *RootDir* directory.

See Also: KeystoreFile

Deleted: OS X

Deleted: OS X

CertificateCN

Sections: [Listener]

Default Value: CDA ITRANS Security

Required: Yes

Description: This will be used to search and retrieve the certificate to configure the REST SSL/TLS communications. It common name of the certificate obtained from the certificate store of the user running CCDWS. This is defined with the CN attribute in the subject field. There should be no reason to change this.

See Also: CertificateIssuer, SSLEnabled

CertificateExportRetries

Sections: [Destination X]

Default Value: 5

Required: No

Description: Specifies the number of retries CCDWS will make when attempting to retrieve a certificate/key pair from the Windows certificate store in case of an error.

See Also: CertificateExportRetrySleep

CertificateExportRetrySleep

Sections: [Destination X]

Default Value: 500

Required: No

Description: Specifies the number of milliseconds CCDWS will sleep before attempting a retry to retrieve a certificate/key pair from the Windows certificate store in case of an error.

See Also: CertificateExportRetries

CertificateFile

Sections: [Destination X]

Default Value: None

Required: No

Description: Specifies the file name for the client certificate/key pair, if SSL/TLS is used in the destination protocol (i.e. if the destination URL begins with https://). This may not be required if client authentication is not used, or if certificates are located in the operating system's certificate store. If an absolute path is not given, the file will be found relative to the destination directory. If the key component of the certificate/key pair is password protected, then the CertificateKeyPassword property is used to decrypt it. Otherwise, ensure that the file is stored in a secure directory. Note that if this option

is provided, it will override the location of certificates in the certificate store using ProviderOidPrefix and OfficeOidPrefix. The contents of CertificateFile are expected to be in PEM format.

See Also: CACertificateFile, CACertificateName, RootCACertificateName, CertificateKeyPassword, ProviderOidPrefix, OfficeOidPrefix, KeyStoreFile, CAKeystoreFile

CertificateFileAction

Sections: [Destination X]

Default Value: Value of Action property

Required: No

Description: If certificate files are being used, then this property will take precedence over that of the Action property. This allows claims using certificate files to have different SOAP actions defined than those using certificates in the certificate store.

See Also: Action, CertificateFile, CertificateFileEndpoint, CertificateFileProviderPrefixes

CertificateFileEndpoint

Sections: [Destination X]

Default Value: Value of Endpoint property

Required: No

Description: If certificate files are being used, then this property will take precedence over that of the Endpoint property. This allows claims using certificate files to have different SOAP actions defined than those using certificates in the certificate store.

See Also: CertificateFile, CertificateFileAction, CertificateFileProviderPrefixes, Endpoint

CertificateFileProviderPrefixes

Sections: [Destination X]

Default Value: None

Required: No

Description: This is a comma-separated list of prefixes that will cause CCDWS to explicitly look to the certificate in the file defined by the CertificateFile property. This allows claims that use the certificate store and those using file-based certificates to monitor the same directory, such as when denturists and/or hygienists use the same instance of CCDWS to submit claims. Setting this property to "8,20" will cause all claims containing provider IDs that begin with an 8 or 20 will use the file defined in CertificateFile, and all others to look in the certificate store. For backward compatibility this is not set, so that if CertificateFile is defined and not CertificateFileProviderPrefixes, all claims will use the certificate file.

See Also: CertificateFile, CertificateFileEndpoint, CertificateFileAction

CertificateIssuer

Sections: [Listener]

Default Value: None

Required: No

Description: Specifies the issuer contained in the listener certificate's subject field. If defined, this will be used to search and retrieve the certificate to configure the REST SSL/TLS communications along with the CertificateCN value. If not defined, then only the CertificateCN value will be used. Valid values are CanDentAssocCA (for production) and CanDentAssocTestCA (for test).

See Also: CertificateCN, SSLEnabled

CertificateKeyPassword

Sections: [Destination X, Listener]

Default Value: None

Required: No

Description: Specifies the password of the key contained in the client certificate/key pair, if the SSL/TLS is used in the destination protocol (i.e. if the destination URL begins with https://). This may not be required if client authentication is not used. If the key is not password protected, then this property is not required. If it is required, ensure that casual users cannot view the configuration file.

See Also: CACertificateFile, CertificateFile, SSLVerify

ConnectTimeout

Sections: [Destination X]

Default Value: 10

Required: No

Description: This property can be set to the number of seconds that a connection attempt will wait before a failure is detected.

See Also: ReadWriteTimeout

CrlRefreshDelay

Sections: [Main]

Default Value: 86400

Required: No

Description: This property can be set to override the default time for downloading a new CRL (certificate revocation list). When the number of seconds between the current time and the last modification time of the CRL stored in the file system exceeds this value, a new one is downloaded during the subsequent transaction. A value of 0 will disable the refresh delay and use only the values of the CRL's nextCRLPublish and nextUpdate properties. Note that these properties will also be used even when CrlRefreshDelay is nonzero, so that either the refresh delay or the CRL properties may trigger an update. The default value is equivalent to one day. If *SslVerify* is set to 0, the CRL will not be downloaded, nor used.

See Also: SslVerify

DetectParseErrors

Sections: [Main], [Destination X]

Default Value: 0

Required: No

Description: CDAnet inputs are parsed to retrieve field values. If this option is set to 1, CCDWS will perform standard validation of the message. If it fails, then it will display details in the log file, as well as setting the output status to 1034 (Request Invalid). CDAnet critical parse failures will still cause a failure and output status of 1034, regardless of the configuration value. Note that payload types other than CDAnet will ignore this option.

EncryptCertificateFile

Sections: [Destination X]

Default Value: None

Required: No

Description: Specifies the name of the certificate file containing the public certificate used to encrypt a message. For either EncryptMessageBody or EncryptMessageParts is to function, EncryptCertificateFile must refer to a valid public certificate file. Additionally WS-Security must also be enabled.

See Also: EncryptMessageParts, EncryptMessageBody, WS-Security

EncryptMessageBody

Sections: [Destination X]

Default Value: 0

Required: No

Description: If EncryptionCertificateFile is configured properly, and WS-Security is enabled, setting this to nonzero will cause the entire message body to be encrypted. Use EncryptMessageParts if the content of specific nodes is to be encrypted.

See Also: EncryptionCertificateFile, WS-Security, EncryptMessageParts

EncryptMessageParts

Sections: [Destination X]

Default Value: None

Required: No

Description: If EncryptionCertificateFile is configured properly, and WS-Security is enabled, setting EncryptMessageParts to a comma separated list of nodes (with associated namespaces) will cause those nodes to be encrypted. For example, “ccd:CDAnet_REQ” will cause the contents of that node to be encrypted. To encrypt the entire body, use EncryptMessageBody.

See Also: EncryptionCertificateFile, WS-Security, EncryptMessageBody.

Endpoint

Sections: [Destination X]

Default Value: None

Required: Yes

Description: The web service requires both an endpoint and an action. The endpoint is specified as a URL (protocol://address/path).

See Also: Action, CertificateFileEndpoint

FaultToAddress

Sections: [Destination X]

Default Value: None

Required: No

Description: If WS-Addressing is enabled, FaultToAddress provides an option to indicate where fault messages should be delivered. This is only necessary if the carrier requires it.

See Also: WS-Addressing, FromAddress, ReplyToAddress

Foreground

Sections: [Main]

Default Value: 0

Required: No

Description: By default, if CCDWS is executed as a Windows application, it will operate in the background by closing its console. If set to 1, it will not close the console. Setting to foreground is useful for testing, when setting LogFile=stdout or LogFile=stderr. This setting has no effect when executed from a user console, or as a service.

FromAddress

Sections: [Destination X]

Default Value: None

Required: No

Description: If WS-Addressing is enabled, FromAddress provides an option to indicate who the message is from. This is only necessary if the carrier requires it.

See Also: WS-Addressing, ReplyToAddress, FaultToAddress

InheritSection

Sections: All

Default Value: None

Required: No

Description: Any section can use this property to define another section to inherit properties from. Properties defined in both the calling section and the inherited section take the value of the calling section (i.e. the specific section value overrides the inherited value). Defining an InheritSection property allows multiple sections to use the same set of property values.

InputFilename

Sections: [Main]

Default Value: input

Required: No

Description: Specifies the base part of the input filename that will be searched for within the destination directories.

See Also: OutputFilename

InstalledComputerName

Sections: [Main]

Default Value: Installed computer name

Required: No

Description: This field is for informational purposes only, and is only populated by the Windows installer. It can be helpful for users to be aware of the computer that certificates must be installed on.

KeystoreFile

Sections: [Main]

Default Value: providerkeys

Required: Yes, for Linux and ~~MacOS~~

Description: For Linux and ~~MacOS~~ installations, specifies the name of the Java keystore file containing provider certificate/key pairs. If no path is given, it is expected to reside in the *RootDir* directory.

See Also: CAKeystoreFile, KeystorePassword

Deleted: OS X

Deleted: OS X

KeystorePassword

Sections: [Main]

Default Value: (exists but not shown here)

Required: Yes, for Linux and ~~MacOS~~

Description: Specifies the password of the Java keystore file containing provider certificate/key pairs, and defined by the *KeyStoreFile* option. Additionally, it is the password for all private keys within the file.

See Also: CAKeystoreFile, KeystoreFile

Deleted: OS X

LogBackupExtension

Sections: [Main], [Listener]

Default Value: .bak

Required: No

Description: Specifies the backup extension to be appended to the log file name when the maximum size of the log file has been reached and it begins a new log file. This option will only be used if the LogMaxSize option is greater than 0 (zero).

See Also: LogMaxSize

LogDir

Sections: [Main], [Listener]

Default Value: value of RootDir

Required: No

Description: Specifies the directory where log files are stored.

See Also: LogFile, LogLevel

LogFile

Sections: [Main], [Listener]

Default Value: ccdws.log in Main section, ccdws_rest.log in Listener section

Required: Yes

Description: The main execution thread and all listening threads write to their own structured log files, with the degree of detail defined by the LogLevel property. If an absolute path is not given, the file will be found relative to the destination directory. Special filenames *stdout* and *stderr* are interpreted to write the file directly to those file descriptors. In a console environment it is easiest to define this property as *stdout* in the global section and not define (or comment out) the properties in the [Main] section. Special substitution variables may also be used:

%Y = century and year

%M = month with leading zero

%D = day of month with leading zero

%h = hour of day with leading zero

%m = minute with leading zero

%s = second with leading zero

If any of these variables changes while log messages are written, the current logfile is closed, and a new one opened with the appropriate new name prior to writing the message.

See Also: LogDir, LogLevel

LogLevel

Sections: [Main], [Listener]

Default Value: 4

Required: No

Description: Specifies the level of detail written to the structured log file. Valid values are: 0 (Off), 1 (Critical), 2 (Errors), 3 (Warnings), 4 (Terse), 5 (Verbose), and 6 (Debug). Each level includes levels below it.

See Also: LogDir, LogFile

LogMaxSize

Sections: [Main], [Listener]

Default Value: 512000

Required: No

Description: Specifies the maximum size of the log file before it will be moved to a backup, so that a new one will begin. If set to 0 (zero), then no maximum size is used. The backup file name will be that of original log file, with a backup extension. This will be .bak by default, and can be controlled by the LogBackupExtension option.

See Also: LogBackupExtension

MaxRetries

Sections: [Main], [Destination X]

Default Value: 1

Required: No

Description: Specifies the maximum number of times CCDWS will attempt to automatically resend a SOAP request. Values configured in Destination sections will override that stored in the Main section of the configuration file. Note that retries will only be attempted for HTTP errors (except 401 and 403, SOAP faults with the actor defined as "Server," read timeouts, or connection failures. Faults returned

from the server with the actor defined as “Client” will not be retried.

OfficeOidPrefix

Sections: [Destination X]

Default Value: 2.16.840.1.113883.3.20.2

Required: No

Description: If set, the default Office OID prefix will be substituted with the new value. This is used in matching the client SSL/TLS certificate in the operating system’s certificate store with the value <OfficeOidPrefix>.<Office ID>, where the value of <Office ID> is extracted from field B02 of the request message. In addition, the value of ProviderOidPrefix behaves in the same way with the Provider ID from field B01 of the request message. Both of these must be satisfied to match the certificate. Note that setting the value of *CertificateFile* will override this matching process and instead use the .PEM formatted file specified.

See Also: OfficeOidPrefix, CertificateFile, CaCertificateFile

OutputFilename

Sections: [Main]

Default Value: output

Required: No

Description: Specifies the base part of the output filename that will be generated in response to an input request.

See Also: InputFilename

Path

Sections: [Listener]

Default Value: /dentalClaim

Required: No

Description: REST communications require a path to be sent in the HTTP header. This should not need to be altered.

See Also: Port

PayloadType

Sections: [Destination X], [Main]

Default Value: CDANET

Required: No

Description: Currently, CDANET is the only payload type that is handled specially, where the input message is parsed and validated, along with other minor features. Other values are supported, but provide no special handling. A PayloadType value provided in the destination section will override the value in the main section. The *Format* node of the resulting SOAP request message will contain this value uppercased. Additionally, for CDANET, it will also automatically append the version number to the *Format* node.

Port

Sections: [Listener]

Default Value: 9000

Required: No

Description: REST communications require a port for the listener to monitor and accept incoming connections. If there is a conflict, then this can be set to another value. Typically, this value will be greater than 1024, as standard ports are defined in the range. However, it is possible that the standard port 80 (HTTP) and port 443 (HTTPS) could be used. Note that 80 is not recommended as this is for clear text communications.

See Also: Path

Prefix

Sections: [Destination X]

Default Value: None

Required: No

Description: This option is specific to CDAnet inputs, for backward compatibility with CCD. If defined, it will overwrite the transaction prefix (A01) field with its value. A special macro %V can be used to substitute the version number of the current input. For example, if the value is CDANET%V, then the prefix will be replaced with CDANET4 for version 4 inputs, and CDANET2 for version 2 inputs. To submit %V without macro substitution, use %%V.

ProviderOidPrefix

Sections: [Destination X]

Default Value: 2.16.840.1.113883.3.20.1

Required: No

Description: If set, the default Provider OID prefix will be substituted with the new value. This is used in matching the client SSL/TLS certificate in the operating system's certificate store with the value <ProviderOidPrefix>.<Provider ID>, where the value of <Provider ID> is extracted from field B01 of the request message. In addition, the value of OfficeOidPrefix behaves in the same way with the office ID from field B02 of the request message. Both of these must be satisfied to match the certificate. Note that setting the value of *CertificateFile* will override this matching process and instead use the .PEM formatted file specified.

See Also: OfficeOidPrefix, CertificateFile, CaCertificateFile

ProxyPassword

Sections: [Main]

Default Value: None

Required: No

Description: If set, then this value will be used for authentication with the proxy server. If so, then the ProxyUserId value should also be set.

See Also: ProxyPort, ProxyServer, ProxyUserId

ProxyPort

Sections: [Main]

Default Value: None

Required: No, unless ProxyServer is defined

Description: If set, then the server will be accessed through the given proxy server address or hostname and port number. If ProxyServer is not defined, then this property is ignored.

See Also: ProxyPassword, ProxyServer, ProxyUserId

ProxyServer

Sections: [Main]

Default Value: None

Required: No

Description: If set, then the server will be accessed through the given proxy address or hostname. Additionally, the ProxyPort option is then required.

See Also: ProxyPort, ProxyServer, ProxyUserId

ProxyUserId

Sections: [Main]

Default Value: None

Required: No

Description: If set, then this value will be used for authentication with the proxy server. If so, then the ProxyPassword value should also be set.

See Also: ProxyPassword, ProxyPort, ProxyServer

ReadWriteTimeout

Sections: [Destination X], [Listener]

Default Value: 60

Required: No

Description: Defines the maximum number of seconds to successfully read from or write to the remote client or to the server. Normally, 60 seconds will be more than ample. If the read or write fails, then an error condition arises and reported in the log file (if LogLevel is set to at least 1), and the session is terminated.

See Also: ConnectTimeout

ReplyToAddress

Sections: [Destination X]

Default Value: None

Required: No

Description: If WS-Addressing is enabled, ReplyAddress provides an option to indicate the address to which the response should be delivered. This is only necessary if the carrier requires it.

See Also: WS-Addressing, FromAddress, FaultToAddress

RootCACertificateName

Sections: [Client X], [Destination X]

Default Value: CanDentAssocROOTCA

Required: No

Description: Specifies the name of the root certificate authority certificate used to sign the CA certificate that generated the client certificate(s) stored in the certificate store. This will be overridden if the value of *CACertificateFile* is provided. This value is ignored if *SSLVerify* is disabled.

See Also: CACertificateName, CACertificateFile, CertificateKeyPassword, KeyStoreFile, KeyStorePassword, SSLVerify

RootDir

Sections: [Main]

Default Value: current working directory of service

Required: No

Description: Specifies the root directory, which should contain the *ccdws.exe* executable, *ccdws.ini* configuration file, and related support files. All destination folders should be subdirectories of this.

See Also: LogDir

SignMessageBody

Sections: [Destination X]

Default Value: 0

Required: No

Description: Setting SignMessageBody to a non-zero value will cause the message body to be signed as part of the WS-Security extension. To sign the entire message, use SignMessageParts instead. Note that WS-Security must be enabled for this option to be used.

See Also: WS-Security, SignMessageParts

SignMessageParts

Sections: [Destination X]

Default Value: None

Required: No

Description: Setting SignMessageParts to a comma separated list of nodes (with associated namespaces) will cause those nodes to be signed. For example, "ccd:CDAnet_REQ, wsa5:To, wsa5:Action" will cause those three nodes to be signed. If EncryptMessageParts is also set to a list of nodes, then the full set of nodes will be signed. To sign just the message body, use SignMessageBody instead. Note that WS-Security must be enabled for this option to be used.

See Also: WS-Security, SignMessageBody, EncryptMessageParts

SSLEnabled

Sections: [Listener]

Default Value: 1

Required: No

Description: By default, if SSL/TLS encryption is used, CCDWS will listen with a certificate that will be used to encrypt communications between the calling client and itself. If set to 0, then the certificate is not configured, and clear text communications are performed. It is recommended that this be done only as a temporary workaround, as it reduces the security of the connection.

See Also: CertificateOid, SSLVerify

SSLVerify

Sections: [Destination X], [Listener]

Default Value: 1

Required: No

Description: By default, if SSL/TLS encryption is used, CCDWS will verify both the local certificate/key pair and the remote server certificate. The verification may be bypassed by setting SSLVerify to 0. It is recommended that this be done only as a temporary workaround, as it reduces the security of the connection.

See Also: CACertificateFile, CertificateFile, CertificateKeyPassword, CRLRefreshDelay

WS-Addressing

Sections: [Destination X]

Default Value: 0

Required: No

Description: Setting WS-Addressing to a non-zero value will enable this web service extension. It will also allow additional WS-Addressing options (FromAddress, ReplyToAddress, FaultToAddress) to be configured.

See Also: WS-Security, FromAddress, ReplyToAddress, FaultToAddress

WS-Security

Sections: [Destination X]

Default Value: 0

Required: No

Description: Setting WS-Security to a non-zero value will enable this web service extension. It will also allow additional WS-Security options (Authentication, SignMessage, SignMessageBody) to be configured.

See Also: WS-Addressing, Authentication, SignMessage, SignMessageBody

7.3. Sample Configuration File

The following *ccdws.ini* configuration file sample shows some common properties in context, along with descriptive comments. Note that all other properties not included will be used with their default values.

```
# =====
#
# Configuration File - Common Communications Driver for Web Services
#
# Configuration filename: ccdws.ini
# Description: Configuration file for CCDWS.
# Created by: Harder Software Ltd.
# Creation Date: 2014-12-01
#
# =====
#
# -----
# Main controller section, containing options for the application
# -----
[Main]

# Levels: 0=off,1=critical,2=error,3=warning,4=terse,5=verbose,6=debug
LogLevel=5

# Base name for input file and output filenames. If CCD is installed, then
# its values will be used. Otherwise the defaults input and output will be
# used.
InputFileName=input
OutputFileName=output

# Uncomment the following line to run in the foreground when executing CCDWS
# as a Windows application. This has no affect when run as a service,
# nor when launched from a user command prompt.
# Foreground=1

# -----
# REST Listener
# -----
[Listener]

Port=9000

# Levels: 0=off,1=critical,2=error,3=warning,4=terse,5=verbose,6=debug
LogLevel=5

# -----
# Common destination section. Each destination section inherits from this,
# but can override in its own section if necessary.
# -----
[Destination Common]

# default time to wait for a successful send or receive. Default of 60 should
# be sufficient.
ReadWriteTimeout=60

# default time to wait for a connection to the remote server. Default of 10
# should be sufficient.
ConnectTimeout=10

# -----
# Alberta Blue Cross
# -----
[Destination ABC]

# Inherit default values from the given section.
```



```

InheritSection=Destination Common

# Value to be overwritten on all transaction prefix (A01) fields, if set.
Prefix=ABCCDANET4

# uncomment the following for testing
Endpoint=https://vendor.claimstream.ca:3542/DWST/HIECService

# uncomment the following line for production
# Endpoint=https://abc.claimstream.ca:3522/DWS/HIECService

# -----
# TELUS Group A
# -----
[Destination TELUSA]

# Inherit default values from the given section.
InheritSection=Destination Common

# Value to be overwritten on all transaction prefix (A01) fields, if set.
Prefix=@

# uncomment the following line for staging
# Endpoint=https://cesdental.uat.telushealth.com/dental/HIEC_vs1

# uncomment the following line for production
Endpoint=https://cesdental.telushealth.com/dental/HIEC_vs1

# -----
# TELUS Group B
# -----
[Destination TELUSB]

# Inherit default values from the given section.
InheritSection=Destination Common

# Value to be overwritten on all transaction prefix (A01) fields, if set.
Prefix=HD*

# uncomment the following line for staging
# Endpoint=https://cesdental.uat.telushealth.com/dental/HIEC_vs1

# uncomment the following line for production
Endpoint=https://cesdental.telushealth.com/dental/HIEC_vs1

```

8. Automated Installation Instructions

These instructions assume that CCDWS will be installed in Windows.

See the section entitled *Manual Installation Instructions* for information on installing CCDWS manually.

8.1. Before Installation

Before installing CCDWS, ensure that any existing CCD or CCDWS run-time environment is known, such as the installation directory (typically c:\ccd). Locations are provided in the *Run-Time Environment* section. The CDAnet certificate locations must also be known. Specifically, note whether they are installed just for the user or whether they were installed at the machine level (making them available to all users).

You will need to have administrative privileges to install CCDWS as a service, so either login as Administrator, or as a user that has these privileges.

Be certain that you have access to the Digital IDs distributed by the Canadian Dental Association. Denturists and

hygienists will not have these, but rather vendor certificates that are stored in the file system.

The latest version will be found at http://www.hardersoft.com/web_updates/ccdws.

8.2. Installing CDA Digital IDs to the Certificate Store

CCDWS supports the storage and retrieval of both client and certificate authority certificates within the operating system's certificate store.

Client certificates distributed by CDA are known as Digital IDs. The Canadian Dental Association provides a special installation application to install its Digital IDs.

In Windows, if CCDWS is installed as a service, it will find certificates that are installed under the local computer context. However, CDA prefers that they be installed under a user account context so they are not available to all users. If this is done, then either the CCDWS must be run as an application under a user that has the certificates installed, or if it is installed as a service, then the service must be configured to log on as a user that has the certificates installed.

Installation and access of certificates in a computer's certificate store is an area that may be prone to configuration issues. If the certificates cannot be found, then there will be connection issues.

8.3. Installing Vendor Certificates

Non-dental providers such as denturists and hygienists do not have Digital IDs available to them, as they are specific to dentists. These providers can make use of file-based certificates that are distributed to vendors by each network. There will typically be one certificate/key pair whose key is password protected, and one CA certificate in common to all vendors. These should be installed in the network destination folder. For instance, if Alberta Blue Cross claims use the c:/ccd/abc directory, then that is where the certificates should be placed.

Once installed, the configuration file must be updated to include appropriate options. See the configuration options *CertificateFile*, *CertificateKeyPassword*, *CACertificateFile*, and *CertificateFileEndpoint*.

If both vendor certificates and Digital IDs are to be used in a single installation (such as if a hygienist shares an office with a dentist), then it is also important to be certain that the *CertificateFileProviderPrefixes* option is configured. Its value would be "8,20" to capture denturists and hygienists. If this is not set, then ALL claims would attempt to use the vendor certificates. If it is set, then only those with provider IDs beginning with these prefixes will use them, and all others will use Digital IDs.

Note that all vendor certificates and related CA certificates should be in .PEM (base64) format, and will normally have a .pem file extension.

8.4. Removing old Installations of CCDWS

As of CCDWS version 1.0, CDAnet certificates are installed into the operating system's certificate store. If an older version is installed, then it must first be uninstalled, or the configuration file will not contain the correct information. You can uninstall it either as an option from the setup application, or from the CCDWS application section of the Windows Start menu.

Note that if you inadvertently uninstall CCDWS, and need to recover the configuration file, it should be found in the installation directory with the filename ccdws.ini.save. It can be copied to ccdws.ini to retain the old configuration.

8.5. Running the Setup Program

The CCDWS setup program is distributed as an executable file. Save it in any accessible location and execute it.

8.6. Accepting the EULA and Reading Installation Notes

When installing for the first time, you will be required to review and accept the End User Licensing Agreement (EULA) before continuing. Once accepted, recent installation notes will be displayed. Please read these notes carefully before continuing. The text of the licensing agreement can be found in section 18.

8.7. Setup Type

When the Setup Type dialog box is displayed, the recommended selection is *Complete*. This will install the CCDWS service and all supported destinations.

Selecting *Compact* will install only the latest CCDWS service. If destinations have previously been installed they will be removed, and their sections in the configuration file will be disabled (not removed).

Selecting *Custom* will allow customized selection of the CCDWS service and all destinations as separate components. At a minimum, for any functionality, the CCDWS service and at least one destination should be installed.

8.8. Destination Folder

The default location for installation of CCDWS is `c:\ccd`, which is the default for the CCD application, and normally the recommended location for CCDWS. Note that if CCD is installed in another location, change the CCDWS location to match it.

8.9. Service Account

Installation of version 1.0 and above will request the name of a service account. Versions prior to this will not.

The CCDWS service must be able to find the certificates it uses for communicating with the remote carriers. If certificates are installed under a user account (this would typically be that of the user performing the installation), then enter the user account with optional preceding Windows domain, and the account password. If certificates are stored under the machine account, then leave the account information blank and it will use the local service account.

8.10. Actions Performed on Installation

During the installation, the appropriate folders will be created, with most necessary files stored in the destination subfolders. The *ccdws.exe* executable and *ccdws.ini* configuration file will be stored in the destination folder.

If a CCD installation already exists, then the installation program will first detect whether CCD is running, and force it to be shut down.

If the CCD configuration file *ccd.ini* contains any network configurations that match those of the supported CCDWS destinations, these will be disabled (but not removed). Note that if CCDWS is uninstalled or a destination is removed, network configurations in *ccd.ini* will be re-enabled.

The CCDWS configuration will extract what it can from *ccd.ini*. For instance, it will use the input and output filename conventions.

8.11. Configuration File Review and Edit

Some things cannot be reliably determined from the existing *ccd.ini* configuration file. Additionally, this may not exist if CCD is not already installed.

Just prior to completion of the installation, and before the CCDWS service is started, a dialog box will display,

asking whether to review or edit the configuration file. Select *Yes* to do so if necessary.

One example of a necessary change to the configuration file is if vendor-specific customizations must be made, such as unique certificate file names, or special CDAnet prefix. This document cannot provide details on the changes that must be made.

8.12. Completion and Starting the CCDWS Service

The final dialog box will contain a checkbox entitled *Run CCDWS now*. This will cause the service to start. In most cases this should remain checked.

8.13. Additional Tasks Upon Successful Installation

If a running instance of CCD was terminated during the installation, then it will need to be restarted.

If vendor or other client-specific certificates are required for any destinations have not been installed into the operating system's certificate store, then they will need to be installed to successfully complete any transactions. Note that the CCDWS service will not need to be restarted if this step is performed after its installation. It will only need to be restarted if the configuration file is subsequently changed.

9. Manual Installation Instructions

These instructions assume that the default file locations are used.

See the section entitled *Automated Installation Instructions* for information on installing CCDWS with a special setup application.

9.1. Before Installation

Before installing CCDWS, ensure that any existing CCD or CCDWS run-time environment is known, such as the installation directory (typically c:\ccd in Windows). Locations are provided in the *Run-Time Environment* section.

In Windows, you will need to have administrative privileges to install CCDWS as a service, so either login as Administrator, or as a user that has these privileges. In Linux and ~~MacOS~~, you must have appropriate read, write, and execute privileges in the CCDWS and related directories.

Deleted: OS X

Be certain that you have access to the Digital IDs distributed by the Canadian Dental Association. Denturists and hygienists will not have these, but rather vendor certificates that are stored in the file system.

The latest version will be found at http://www.hardersoft.com/web_updates/ccdws.

9.2. Extracting Files

If an existing CCDWS service instance is running, ensure that it's stopped, as per the section entitled *Controlling the CCDWS Service*.

The provided "zip" file will contain the latest application and configuration files in the appropriate directories for installation. Assuming the CCDWS home directory is c:\ccd, and the zip file is named ccdws_1001.zip, copy the zip file into a temporary location, and open it within Windows Explorer or the Mac Finder. You should see a single directory *ccd* within that archive. Drag this with your mouse to drive C in Windows, or the appropriate installation directory in Linux and ~~MacOS~~.

Deleted: OS X

If files already exist, you may be prompted to replace them. It is best to back up the existing directory before doing so.

9.3. Installing CDA Digital IDs to the Certificate Store or Keystore

CCDWS supports the storage and retrieval of both client and certificate authority certificates within the operating system's certificate store.

The Canadian Dental Association provides a special installation application to install its certificates.

In Windows, if CCDWS is installed as a service, it will find certificates that are installed under the local computer context. However, CDA prefers that they be installed under a user account context so they are not available to all users. If this is done, then either the CCDWS must be run as an application under a user that has the certificates installed, or if it is installed as a service, then the service must be configured to log on as a user that has the certificates installed.

In Linux and MacOS, a *providerkeys* and *cacerts* keystore files are created in the current working directory if they do not exist. If they do exist, ensure that they are placed in the currently working directory prior to running the installation application. The two keystores generated can either be placed in the CCDWS root directory or in some other location if specified in the configuration file.

Deleted: OS X

Installation and access of certificates in a computer's certificate store is an area that may be prone to configuration issues. If the certificates cannot be found, then there will be connection issues.

9.4. Installing Vendor Certificates

See section 8.3. for information concerning the installation of vendor certificates.

9.5. Updating the CCDWS Configuration File

The base configuration file provided will be called *ccdws_default.ini*. This prevents it from overwriting any customized configuration you may already have, such as proxy information, etc.

If this is the first installation, then copy *ccdws_default.ini* to *ccdws.ini*. Otherwise, you should do a manual comparison and merge any changes to the existing *ccdws.ini* file. Use a text editor such as Notepad to perform these changes.

For each destination XYZ, ensure a section called [Destination XYZ] exists. For instance, ensure that the section [Destination ABC] exists if the ABC (Alberta Blue Cross) subdirectory exists, or [Destination TELUSA] exists if the TELUSA (TELUS) subdirectory exists. The destination section name is case insensitive. You can find contents for these sections in the ccd subdirectories (i.e. abc, telus).

9.6. Updating the Existing CCD Configuration File

If the CCD application is already installed and is configured to submit claims to the same destination, its configuration file will need to be altered so that it does not attempt to send claims to the same destination as CCDWS. Use a text editor such as Notepad to view and alter the file *c:\ccd.ini*.

If a section entitled [XYZ] exists in the *ccd.ini*, change the section name. The automated installation application will change the name to [XYZ_DISABLED].

If the CCD configuration file is an older style, it may contain a line such as *network XYZ*, followed by a line containing the word *begin*. In this case, change the line *network XYZ* to some alternative name. The automated installation application will change this to *network XYZ_DISABLED*.

Restart CCD when this has been completed.

9.7. Confirming CCDWS Operation

Once everything is properly configured, it should be confirmed that the CCDWS can run within the operating

environment. To do so, open up a command prompt, and enter the following commands:

```
cd c:\ccd  
ccdws -v
```

The output from the second command should look something like the following (with expected differences in version and creation date:

```
CCDWS - Common Communication Driver for Web Services  
Version 1.0.6.0 built Jul 31 2017 09:05:48  
Copyright (C) 2011-2017 CLHIA / Harder Software Ltd.
```

9.8. Installing the CCDWS Service

In Windows, if this is the first time that CCDWS is being installed, it must also be installed as a Windows Service. If it has been installed as a service previously, then this step need not be performed.

To install as a service, open a Windows command prompt, and enter the following commands, each following by the enter key:

```
cd c:\ccd  
ccdws -i
```

The output from the second command should look the following:

```
CCDWS service installed.
```

This will install the service, but it will not yet be running. To start and control the service, see the following section entitled *Controlling the CCDWS Service*.

In Linux and MacOS, CCDWS can be run as a service, as these are simply command-line applications. Typically this is done with a special service control script. This is not yet included with CCDWS.

Deleted: OS X

10. Controlling the CCDWS Service

CCDWS will be normally run as a service, so that it will start automatically when the operating system starts, and stop when the operating system is shut down. When it is first installed it will not yet be running until it is either started manually, or the operating system is restarted.

These instructions are specific to Windows. Note that the user that controls CCDWS should have Administrative control of the host, or control of service startup and shutdown.

10.1. GUI Control

In the Windows environment, CCDWS execution is controlled through the Windows Services. To access the services dialog box, from the Start menu, go to Control Panel -> Administrative Tools -> Services. The CCDWS service can be found by scrolling through the service names.

The service is controlled by right clicking on the CCDWS service name, and selecting *Start* or *Stop*. To control whether the service start automatically at runtime, select *Properties* and change the *Startup type*. Setting to Manual causes the service to only start manually. Setting to Automatic will cause it to start when Windows starts. Manual control of the service is also possible from this dialog, using the *Start* and *Stop* buttons. *Pause* and *Resume* are not enabled.

10.2. Command Line Control

Windows also has a command line tool for controlling services, called `sc`. This may be executed from a host other than the one where CCDWS is installed. If so, then supply the hostname preceded by two backslashes (e.g. `\\hostname`). In the following examples, replace `[hostname]` with the correct hostname in the format given. Otherwise, leave the argument out of the command.

To check the status of the service, enter the following:

```
sc [hostname] interrogate ccdws
```

To start the service, enter the following:

```
sc [hostname] start ccdws
```

To stop the service, enter the following:

```
sc [hostname] stop ccdws
```

In Linux and [MacOS](#), control of the service is dependent on the script used. This is not currently provided with CCDWS.

Deleted: OS X

11. Executing CCDWS from the Command Line

Although CCDWS will be normally run as a service, it can be executed from the command line. This is useful mainly for troubleshooting. To do so in Windows, open a command prompt and enter the following commands (assuming CCDWS is installed in the default location):

```
cd c:\ccd  
ccdws
```

In Linux and [MacOS](#), the location will be wherever CCDWS is installed. For instance:

```
cd /usr/local/sbin  
./ccdws
```

Deleted: OS X

In Windows, the application will continue to run in the foreground until the command prompt is closed, or `Ctrl-C` is pressed. In Linux and [MacOS](#), it will immediately run in the background with no log messages sent to the screen unless the *ForegroundMode* configuration option is set.

Deleted: OS X

From the command line it is possible to display log information directly to the console, which is very useful when debugging. To do this, set the *LogFile* configuration value to either *stdout* or *stderr*. The *LogLevel* value should be set to see the desired level of information.

12. Log Files

CCDWS writes to log files to record information relating to connections, hex dumps of messages sent and received, warnings, errors, and debugging information. The amount of detail included is determined by the log level that is configured.

12.1. Log File Format

Format of the log files is quite structured, with the following features:

- Each time CCDWS is started, a header is written to the log indicating the application name and version.

The time the log opened is also recorded.

- ❑ Each time CCDWS is stopped, a footer is written indicating the time the log was closed.
- ❑ Every entry is preceded by a timestamp in the form YYYY-MM-DD HH:MM:SS>.
- ❑ Verbose and Debug levels show multi-line message contents as hexadecimal dumps.

12.2. Configuring Log Filenames and Levels

The configuration setting *LogDir* should be set to the location where all of the log files will reside. This defaults to the *RootDir* directory (The current working directory in Windows and MacOS, and */var/spool/ccd* in Linux).

If the *LogFile* configuration value is a simple filename or relative path then that will be appended to the *LogDir* value for a complete path. If the value begins with a slash or backslash (i.e. '/' or '\') or with a drive letter, then it is considered an absolute path and the *LogDir* value is not prepended to it. Additionally, if the *LogFile* value is *stdout* or *stderr*, then the *LogDir* value is ignored and the log file will be written to either the *stdout* or *stderr* file handle. Note that the latter case will only work when CCDWS is executed from the command line mode. The default value for *LogFile* is *ccdws.log*.

The amount of information stored in the log is configurable. See the *LogLevel* property in the *Configuration Reference*.

Note that each level also records lower level messages. For example, the Terse level (4) also records Critical (1), Error (2) and Warning (3) messages.

0 – Off: No messages at all are recorded.

1 – Critical: Errors causing the application to fail

2 – Errors: Only error messages are recorded.

3 – Warnings: Non-critical warning messages are recorded. It is good practice to repair the source of warnings when these messages appear.

4 – Terse: Small amount of information recorded.

5 – Verbose: Additional informational messages, plus hexadecimal dumps of request and response messages. Use this level to record full transaction information.

6 – Debug: Internal method calls and contents of variables are recorded. This level should only be used during development or when troubleshooting problems.

NOTE: While CCDWS supports these levels, there are currently no Critical messages defined.

12.3. Log File Rotation

Log files can be automatically rotated by using any of the substitution variables as defined in the *LogFile* property of the *Configuration Reference* section. For instance, %Y%M%D embedded in the name will cause the filename to change daily, assuming messages are written to the log.

Alternatively, the *LogMaxSize* property in the configuration file can be used to limit the size of log files. When the log file size passes this size, it is moved to a backup file with the extension given by *LogBackupExtension* (the default is *.bak*), and the original file is truncated. The default value for *LogMaxSize* is 512000 (500 KB).

If *LogMaxSize* is set to 0 (i.e. disabled) and substitution variables not used in *LogFile*, then the log file should be watched and either rotated manually or with a separate application. Otherwise, it will continue to grow without bound.

12.4. Sample Log File with Notes

The following log file was generated from a test server with LogLevel = 5 (verbose). It has been condensed for clarity. Notes follow the end of the log.

```
=====
CCDWS (beta) version 0.9.0.24
Log opened 2011-10-25 09:08:39
=====
2011-10-25 09:09:10> Found 1 request.
2011-10-25 09:09:10> Read 500 bytes from input file c:\ccd\ABC\input.001.
000000: 41 42 43 74 65 73 74 20-20 20 20 20 30 30 30 30 ABCtest 0000
000010: 30 34 30 34 30 31 30 30-30 30 39 30 48 31 20 31 040401000090H1 1
000020: 30 30 35 30 30 20 30 30-30 30 34 35 39 30 31 32 00500 0000459012
000030: 32 34 30 30 34 34 36 39-35 39 30 31 32 32 34 30 2400446959012240
000040: 30 34 34 36 39 20 20 20-20 20 20 20 20 20 20 30 04469 0
...
0001B0: 30 30 30 30 30 58 20 20-20 20 30 30 32 30 32 31 00000X 002021
0001C0: 31 31 32 30 31 31 31 30-32 34 30 30 20 20 20 20 112011102400
0001D0: 20 30 30 31 33 33 30 20-20 20 20 20 30 30 30 30 001330 0000
0001E0: 30 30 20 20 20 20 20 30-30 30 30 30 30 58 20 20 00 000000X
0001F0: 20 20 30 30 00 00
2011-10-25 09:09:11> Sending 1367 bytes to destination ABC.
000000: 50 4F 53 54 20 2F 44 57-53 31 2F 43 44 41 6E 65 POST /DWS1/CDAn
000010: 74 53 65 72 76 69 63 65-20 48 54 54 50 2F 31 2E tService HTTP/1.
000020: 31 0D 0A 48 6F 73 74 3A-20 68 69 65 63 2E 61 62 1..Host: hiec.ab
000030: 2E 62 6C 75 65 63 72 6F-73 73 2E 63 61 0D 0A 55 .bluecross.ca..U
000040: 73 65 72 2D 41 67 65 6E-74 3A 20 67 53 4F 41 50 ser-Agent: gSOAP
...
000510: 20 20 20 20 20 30 30 30-30 30 30 58 20 20 20 20 000000X
000520: 30 30 3C 2F 63 63 64 3A-43 44 41 6E 65 74 5F 52 00</ccd:CDAnet_R
000530: 45 51 3E 3C 2F 53 4F 41-50 2D 45 4E 56 3A 42 6F EQ></SOAP-ENV:Bo
000540: 64 79 3E 3C 2F 53 4F 41-50 2D 45 4E 56 3A 45 6E dy></SOAP-ENV:En
000550: 76 65 6C 6F 70 65 3E velope>
2011-10-25 09:09:22> Received 1510 bytes from destination ABC.
000000: 48 54 54 50 2F 31 2E 31-20 32 30 30 20 4F 4B 0D HTTP/1.1 200 OK.
000010: 0A 44 61 74 65 3A 20 54-75 65 2C 20 32 35 20 4F .Date: Tue, 25 O
000020: 63 74 20 32 30 31 31 20-31 36 3A 30 39 3A 31 31 ct 2011 16:09:11
000030: 20 47 4D 54 0D 0A 53 65-72 76 65 72 3A 20 4F 72 GMT..Server: Or
000040: 61 63 6C 65 2D 41 70 70-6C 69 63 61 74 69 6F 6E acle-Application
...
0005A0: 64 65 72 61 74 69 6F 6E-2E 20 20 20 20 20 20 20 deration.
0005B0: 20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
0005C0: 20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
0005D0: 20 20 20 20 20 20 20 20-20 20 3C 2F 43 44 41 6E </CDAn
0005E0: 65 74 5F 52 45 53 et_RES
2011-10-25 09:09:22> Received 24 bytes from destination ABC.
000000: 50 3E 3C 2F 53 3A 42 6F-64 79 3E 3C 2F 53 3A 45 P></S:Body></S:E
000010: 6E 76 65 6C 6F 70 65 3E nvelope>
2011-10-25 09:09:22> Received 2 bytes from destination ABC.
000000: 0D 0A ..
2011-10-25 09:09:22> Received 5 bytes from destination ABC.
000000: 30 0D 0A 0D 0A 0....
2011-10-25 09:09:22> Wrote 580 bytes to output file c:\ccd\ABC\output.001.
000000: 34 2C 30 2C 30 30 31 2C-41 42 43 74 65 73 74 20 4,0,001,ABCtest
000010: 20 20 20 20 30 30 30 30-30 34 30 34 32 31 30 30 000004042100
000020: 30 30 39 30 30 30 35 37-31 59 35 39 30 31 32 32 009000571Y590122
000030: 34 30 30 34 34 36 39 33-33 33 33 36 39 37 31 20 400446933336971
000040: 20 20 20 20 20 30 30 30-30 30 30 30 30 30 30 30 000000000000
...
000200: 66 6F 72 20 63 6F 6E 73-69 64 65 72 61 74 69 6F for consideratio
000210: 6E 2E 20 20 20 20 20 20-20 20 20 20 20 20 20 20 n.
000220: 20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
000230: 20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
000240: 20 20 20 0D .
```

```
2011-10-25 09:09:22> Processed 1 transaction: 1 succeeded, 0 failed.
2011-10-25 09:11:53> Signal 2 received.
```

```
-----
Log closed 2011-10-25 09:11:53
-----
```

Notes regarding the log:

- ❑ The CCDWS service was started for a single transaction then stopped.
- ❑ A single input was discovered, for the destination ABC. Its filename was input.001.
- ❑ Contents of the messages sent and received are shown in hexadecimal on the left and ASCII on the right. These are called 'hex dumps'.
- ❑ Because of the large size of XML messages, ellipses (...) are used between the first five and last five lines of each hex dump section.
- ❑ The input request was 500 bytes long. This can be seen both from the given log message, and from the actual hex dump numbers. Note the left side of the last line of the input hex dump is 0x0001F0, indicating the position of the first byte. Following it are three more bytes, so that last byte number is 0x0001F3. It is a zero offset count, so add one more to the bytes position to get 0x0001F4. Translated to decimal, this is 500.
- ❑ The input was translated to a SOAP request message, transported over HTTP. The total size of the transported message was 1367 bytes.
- ❑ CCDWS successfully connected to the endpoint <https://hiee.ab.bluecross.ca/DWS1/CDAnetService>, as shown in the HTTP header. The https is not shown as part of the HTTP protocol, but was contained in the actual message's WS-Addressing nodes.
- ❑ A chunked response (1510, 24, 2, 5 bytes) from the server was received, containing the SOAP response.
- ❑ The output file output.001 of size 580 bytes was written to disk. From the first header fields of that file, we can see that the office sequence number was 4, the status was 0 (success), and the file extension was 001. With comma separators these make up a total of 8 bytes, with a since 0xD byte terminator. So, the actual CDAnet response message was $580 - 8 - 1 = 571$ bytes in size.
- ❑ CCDWS determined that this was a success since it received the response, regardless of the status of the actual response, so it indicated that one transaction was processed, one succeeded and none failed.
- ❑ Overall time for the transaction was approx 12 seconds, starting at 09:09:10 and ending at 09:09:22. The majority of this time was waiting for a response, as we can see that the request was sent at 09:09:11, and response was received at 09:09:22. This is a relatively long time

13. Status Codes

CCDWS uses a subset of CCD status codes to populate the output file. Many of the CCD codes do not apply as they are specific to telephony communications. For errors that don't fit within the CCD code set, the default 1001 code will be returned. For error conditions, further error details including internal error codes are recorded in the log file.

Code	Message	Description
0	Success	The request was sent to the remote server and the response was successfully received and stored in the output file.

Code	Message	Description
1001	General error	A general error occurred. Check the log file for details, including internal error
1026	No answer	CCDWS could not connect to the remote server.
1033	Error reading input	The input file could not be read.
1034	Request invalid	The input request does not conform to CDAnet v2, v3, or v4 message standards. See the log file for details.
1042	Server timeout	A timeout occurred either sending the message to the server (write timeout) or receiving a response from the server (read timeout).
1043	Invalid characters	The server responded with unexpected data. This may happen in the server address is incorrect or the server is not correctly configured.
1045	Server disconnect	The server disconnected unexpectedly. This will most frequently occur if a connection was made, but an SSL/TLS error was detected.

14. Internal Error Codes

The following subsections describe errors that may appear in log files, along with descriptions (and possible resolutions where appropriate).

Numbers below 6000 or about 10000 may vary on different operating systems. Here they refer to Windows error values.

Code	Message	Description
2	The system cannot find the file specified.	The specified file does not exist or cannot be found. This message can occur whenever a specified file does not exist
3	The system cannot find the path specified.	The specified directory does not exist, or a component of a path does not specify an existing directory.
4	The system cannot open the file.	No more file descriptors are available, so no more files can be opened.
5	Access is denied.	Application may not have permissions to read, write, or execute one or more files, such as a log file, configuration file etc. Ensure that CCDWS user has rights to all its read and write locations.
8	Not enough storage is available to process this command.	Not enough memory on CCDWS host, or there is a memory leak. Try rebooting workstation. Contact support personnel if the problem persists.

Code	Message	Description
112	There is not enough space on the disk.	Log files and/or temporary files cannot be written because there is no space left on the hard disk. Free up space by deleting old files, check the disk partitions to be certain that it is correctly partitioned, and look into purchasing a larger disk if necessary.
6012	Payload parse error.	Payload contents were not in the expected format. Remote client is sending invalid data.
7001	SOAP: The service returned a client fault.	A fault occurred, and the server has determined it is a problem with CCDWS. Ensure the configuration for the destination is correct.
7002	SOAP: The service returned a server fault.	A fault occurred, and the server has determined it is a problem with the server itself. If the problem continues contact support personnel.
7011	SOAP: Internal error.	Contact support personnel.
7012	SOAP: An exception raised by the service	General SOAP fault. Neither client nor server has been specified. Look for further information in the log file.
7014	SOAP: No data in HTTP message.	An HTTP response was returned but did not contain a body.
7020	SOAP: Out of memory.	Not enough memory on CCDWS to create SOAP objects, or there is a memory leak. If host has sufficient memory ensure other memory intensive applications are not using it up. Try rebooting workstation. Contact support personnel if the problem persists.
7023	SOAP: An element was null while it is not supposed to be null.	The WSDL expects a non-null element, but a null element was found. Contact support personnel.
7028	SOAP: A connection error occurred.	The endpoint is not specified in the configuration file or the destination host is not accepting connections on the expected port. Confirm the endpoint with the destination administrator. Ensure firewall is allowing access to the endpoint.
7030	SOAP: An SSL error occurred.	A general SSL occurred. Details may be displayed a fault message in the log.
7039	SOAP: SOAP version mismatch or no SOAP message.	Check that the endpoint for the destination refers to a webservice destination. HTTP responses that contain data but not a SOAP response will generate this error.
7098	SOAP: Read timeout.	A timeout occurred waiting for a response. Contact support personnel if the problem persists.
7099	SOAP: Server disconnect.	This can occur if the destination server terminated the connection. If the error continues contact support personnel.

Code	Message	Description
7400	HTTP: Bad Request.	The web service appears to be malformed. Check that all of the configuration options for the destination are appropriate.
7401	HTTP: Unauthorized.	The endpoint attempting to be accessed requires authentication. Check that all of the configuration options for the destination are appropriate.
7403	HTTP: Forbidden.	The server has forbidden access to the endpoint attempting to be accessed. Check that all of the configuration options for the destination are appropriate.
7404	HTTP: Not Found.	The endpoint's path is not valid. Check that all of the configuration options for the destination are appropriate.
7405	HTTP: Method Not Allowed.	Some servers are poorly configured where the endpoint is a path or is to be redirected. If it is specified with a trailing slash, ensure that this is provided on the endpoint.
7407	HTTP: Proxy Authentication Required.	Ensure that the ProxyUserId and ProxyPassword configuration options are populated correctly, and that all other proxy related options are valid.
7408	HTTP: Request Timeout.	Problem writing data to the destination host. Try again, and contact support personnel if the problem persists.
7415	HTTP: Unsupported Media Type.	This may be a result of an incompatibility between the SOAP version submitted by CCDWS and that of the destination server. Contact support personnel.
7500	HTTP: Internal Server Error.	An internal error occurred at the destination server. Contact support personnel.
7501	HTTP: Not Implemented.	The destination server does not support the functionality to fulfil the request. Confirm that the destination endpoint is correct. Contact support personnel if the problem persists.
7502	HTTP: Bad Gateway.	The destination server received an invalid response from an upstream server. Contact support personnel if the problem persists.
7503	HTTP: Service Unavailable.	The destination server is unable to handle the request due to temporary overloading or maintenance of the server. Contact support personnel if the problem persists.
7504	HTTP: Gateway Timeout.	The destination server did not receive a timely response from an upstream server. Contact support personnel if the problem persists.
7505	HTTP: HTTP Version not supported.	The SOAP version supported by CCDWS and that of the destination are not compatible. Contact support personnel.

15. Troubleshooting

If problems occur in the operation of CCDWS, it is most useful to increase the value of LogLevel to 6 in the *ccdws.ini* configuration file, and restart the service. Doing so provides the maximum amount of information in the log file. Often it is enough for a vendor or other technical person to review the log and identify the issue. Otherwise, the log file or relevant portions can be sent to support personnel for analysis.

The following table can be used to help troubleshoot common issues that may arise during CCDWS execution:

Status Code	Internal Error Code	Problem	Resolution
1001	5	CCDWS is denied access to write a file. Confirmed by a message in the log file.	This error may be due to permission issues of the CCDWS service. It must be able to write to both its root directory and all of its subdirectories. Check that the appropriate permission is provided. If system policies prevent the local service user from writing to these directories, set the service user as a domain user with the necessary privileges.
1001	7030	SSL handshake error. Confirmed by message in log indicating certificate for provider ID and office could not be found and SSL tcp_connect error.	Check that certificate that matches both the provider and office in the input message is installed in the certificate store, and is accessible by the user CCDWS is operating under.
1001	7030	SSL verify error of remote endpoint. Confirmed by message in log also showing certificate issuer and subject.	Check that CACertificateFile configuration option in ccdws.ini is NOT set if using latest CDA certificates. Check the Endpoint configuration option in ccdws.ini is correct for the destination. Check that CanDentAssocCA and CanDentAssocROOTCA certificates are both installed in the operating certificate's certificate store. The former should be an intermediate CA, while the latter should be a ROOT CA.
1001	None	The CRL file downloaded is not in the correct format. Check for an error in the log file for a message indicating the CRL file could not be loaded. Prior to that, there may be a line in the log indicating an ASN1 tag error.	Delete all *.CRL files in the CCDWS root directory. If the error continues with subsequent claims then SSL verification can be temporarily disabled using the SSLVerify option.

Status Code	Internal Error Code	Problem	Resolution
1001	None	The certificate corresponding to the provider identified in the transaction is in the CRL. The log file will contain a line indicating that the certificate is in the CRL and the transaction will not be attempted.	Contact CDAnet to determine why the certificate is in the CRL. Obtain and install a new certificate if appropriate.
1001	None	CCDWS cannot find a valid certificate for the dentist even though the Digital ID has been correctly installed. It is attempting to use a vendor certificate to when connecting.	Ensure that CertificateFileProviderPrefixes configuration option is correctly defined. This will ensure that only non-dental providers use the vendor certificates. Otherwise, if CertificateFile is defined, then all claims will attempt to use that file.
1013	None	The appropriate certificate/key pair could be found for the current transaction, but the key is marked as non-exportable, so that transaction failed. The log file will contain a message indicating the key was marked as non-exportable.	Check whether the key is marked as non-exportable by viewing the certificate in the certificate store. If it is not exportable, the option to export the key with the certificate will be disabled. You must then either delete and install the certificate marking the key as exportable, or export the certificate/key pair using a third-party tool such as Jailbreak, then re-import with the key marked as exportable.
1013	None	A valid certificate could not be found for the dental provider and office. This can mean the certificate does not exist, or its date range is not valid for the current date (typically this means the certificate is expired). Confirm by checking the log file for an error indicating that a valid certificate could not be found.	Contact CDAnet to obtain a valid certificate.
1045	7030	The CRL is out of date. Confirmed by checking the log file for a line indicating an SSL verify error with the reason that CRL has expired.	Delete the existing *.CRL file (or files) in the CCDWS root directory and rerun the service. Ensure that the date and time is correct on the computer where CCDWS is installed. If not, set the time and take steps to ensure the time stays current. Ensure that the CCDWS root directory is writable by the CCDWS service.

Status Code	Internal Error Code	Problem	Resolution
1045	7030	The server endpoint is not correct for the certificate being used. sha1 and sha256 certificates must use different endpoints. Confirmed by two messages in the log file. The first indicates "unable to get certificate CRL". The second contains indicates that the certificate verify failed. Additional confirmation can be found by looking for the signature hash algorithm identified in the log file. For Windows XP hosts, this should be sha1WithRSAEncryption. For all others it should be sha256WithRSAEncryption.	Review certificates in the Windows certificate store, and verify that the signature hash algorithm matches that required for the operating system. If multiple certificates are in the certificate store for the same provider, then delete those that do not have the correct hash algorithm. If only one exists, then contact CDAnet to obtain a new certificate as it may have been installed incorrectly, or may be an artifact from an operating system upgrade. If the signature hash algorithm is correct for the operating system, then the server's endpoint may be incorrect in the configuration file. Verify this with the carrier to ensure that the correct endpoint for the operating system is configured.

16. OpenSSL License

CCDWS uses binaries from the OpenSSL project to provide support for SSL encrypted connections. The OpenSSL licence is included below:

Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above **copyright** notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above **copyright** notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eyay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eyay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eyay@cryptsoft.com). The implementation was written so as to conform with Netscape SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes cryptographic software written by Eric Young (eyay@cryptsoft.com)"

The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgment:

"This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

17. gSOAP License

Harder Software uses gSOAP tools and code to support web service development. It has a commercial license to generate commercial code using the gSOAP code generation tools. Applicable gSOAP source code is bundled with the application source code. See the gSOAP public license at <http://www.cs.fsu.edu/%7eengelen/license.pdf>.

18. CCDWS End User Licensing Agreement (EULA)

In order to use CCDWS, users must agree and comply with the following licensing agreement. This will be also be displaying when installing the software:

END USER LICENSING AGREEMENT

The Canadian Life and Health Association ("CLHIA"), the owner of the Common Communications Driver – Web Services (CCDWS, the "Software"), is willing to grant you, or, in the case that you represent a corporation or other organization, that corporation or organization (collectively and interchangeably, "Licensee" or "You") a limited, personal, non-exclusive license to use the Software subject to Your acceptance and agreement to be bound by the terms of this End User Software License Agreement ("Agreement").

By installing/using this Software, You acknowledge that You have read, understand, and agree to be bound by the terms of this Agreement as it relates to this software as of the date on which you first click the "Accept" button. If You do not click "Accept" to the terms of this Agreement, CLHIA is unwilling to grant You a license to the Software.

1. Grant of License. Subject to the terms and conditions of this Agreement, CLHIA grants to You a personal, limited, non-exclusive, non-transferable license to use the Software.

2. Installation. You may install, use, access, display and run the Software on any number of computers, such as workstations, web servers or other devices ("Workstations"). You may also store or install the Software on a storage device, such as a network server used to install or run the Software on Your other Workstations over an internal network.

3. Restrictions. Except as expressly permitted under this Agreement, You will not, nor will You allow any third party, to: (a) modify, translate, adapt, alter, reverse engineer or attempt to extract the source code, or create derivative works from the Software; (b) merge the Software with any other software or documentation; or, (c) sell, sublicense, rent, or lease the Software to any other third party. Further, You may not remove, alter or obscure any proprietary notice that appears on the Software or on any copies made in accordance with this Agreement.

4. Ownership. The Software is licensed, not sold, to You for use solely subject to the terms and conditions of this Agreement. The Software and all worldwide intellectual property and proprietary rights therein and relating thereto, are and will remain the exclusive property of CLHIA or its licensors. Except for the limited rights expressly granted under Clause 1 herein, You will have no right, title or interest (whether by implication, estoppel, or otherwise) in or to the Software or any Intellectual Property Rights (as defined in this clause) therein or thereto. CLHIA retains all rights, title and interest in and to any and all trademarks and logos of CLHIA displayed on or in the Software. You agree not to challenge or contest CLHIA's rights to or ownership of, or otherwise attempt to assert any rights in, the Software. "Intellectual Property Rights" means all worldwide patent, patent applications, copyrights, trade secrets, trademarks, service marks, trade names and any other intellectual property, proprietary, and database protection rights.

5. Third Party Code. The Software may contain or include software code owned or provided by third-party licensors of CLHIA ("Third-Party Code"). For any Third-Party Code clearly indicated to be subject to the terms of a third party software license (a "Third-Party License"), the terms of the applicable Third-Party License will apply to the Third-Party Code independent of the terms of this Agreement. Any Third Party Code not subject to a Third Party License is subject to the terms and conditions of this Agreement and the licensors of any such Third Party Code are third party beneficiaries of this Agreement. Nothing in this Agreement limits Your rights under, or grants rights to You that supersede, the terms of any applicable Third-Party License.

6. Jurisdiction. This Agreement shall be governed by and construed in accordance with the laws of the Province of Ontario and the laws of Canada and the parties attorn to the jurisdiction of the courts of the Province of Ontario.

7. License Fee. There will be no license fee associated with the Software. However, You will be responsible for all costs associated with implementation of the Software for Your purposes.

8. Term and Termination. The term of this Agreement will begin upon Your installation and or use of the Software and, unless earlier terminated as set forth in this Agreement, will continue indefinitely. You may terminate this Agreement at any time by providing notice to CLHIA. This Agreement will also automatically terminate if You breach a material term of this Agreement. Upon any termination of this Agreement, You agree to immediately cease all use of the Software, destroy all copies of the Software, and, upon the request of CLHIA, certify in writing Your compliance with the terms and conditions of this Section 8. Sections 4, 9, 10, 11, and 12, shall survive termination of this Agreement.

9. Warranty and Disclaimer. CLHIA DOES NOT WARRANT THAT: (A) THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE OR THAT FUNCTIONS CONTAINED IN THE SOFTWARE WILL OPERATE IN COMBINATIONS OF SOFTWARE OR HARDWARE THAT MAY BE SELECTED FOR USE BY YOU; (B) THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR EXPECTATIONS; OR (C) ANY RESULTS, OUTPUT, OR DATA PROVIDED THROUGH OR GENERATED BY THE SOFTWARE WILL BE ACCURATE, UP-TO-DATE, COMPLETE OR RELIABLE. CLHIA DOES NOT REPRESENT OR WARRANT THAT THE SOFTWARE WILL BE UP TO DATE, MAINTAINED OR AVAILABLE AT ANY POINT IN TIME. THE SOFTWARE IS PROVIDED STRICTLY ON AN AS IS WHERE IS BASIS. EXCEPT

AS EXPRESSLY STATED IN THIS SECTION, TO THE MAXIMUM EXTENT PERMITTED BY LAW, CLHIA SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, ORAL OR WRITTEN, ARISING BY LAW OR OTHERWISE, RELATING TO THIS AGREEMENT AND THE SOFTWARE AND ANY SERVICES PROVIDED TO YOU, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

10. Limitation of Liability. IN NO EVENT WILL CLHIA BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED, WHETHER FOR BREACH OF CONTRACT, BREACH OF PRIVACY, NEGLIGENCE OR OTHERWISE, AND REGARDLESS OF WHETHER CLHIA HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, INCLUDING WITHOUT LIMITATION, THE USE OR INABILITY TO USE THE SOFTWARE, OR ANY RESULTS OBTAINED FROM OR THROUGH THE SOFTWARE. CLHIA WILL NOT BE LIABLE FOR ANY NETWORK-RELATED PROBLEMS ATTRIBUTABLE TO THE SOFTWARE OR CHANGES TO NETWORK CONFIGURATION THAT MAY AFFECT THE PERFORMANCE OF THE SOFTWARE.

11. Indemnification. You will indemnify, defend, and hold harmless CLHIA, its licensors, and each of their respective employees, officers, directors, and affiliates ("Indemnified Parties"), from any and all claims, losses, liabilities, damages, fees, expenses and costs (including attorneys' fees, court costs, damage awards, and settlement amounts) which result from any claim or allegation against any Indemnified Party arising from Your use of the Software or Your breach of any term of this Agreement. CLHIA will provide You with notice of any such claim or allegation, and CLHIA will have the right to participate in the defence of any such claim at its expense.

12. Confidential Information. You acknowledge that the Software contains confidential and proprietary information of CLHIA, including without limitation the Source Code, inventions, algorithms, knowhow and other proprietary information contained therein (collectively, "Confidential Information"). You agree to protect the Confidential Information with at least the same degree of care employed with respect to Your own confidential or proprietary information. You will not use the Confidential Information for any purpose other than in connection with Your use of the Software under the Agreement. Except as otherwise set forth in this Agreement, under no circumstances will You allow any third party to have access to the Software.

13. Assignment. You may not assign, delegate or otherwise transfer this Agreement or any of Your rights or obligations under this Agreement without the prior written consent of CLHIA. Unless specifically authorized in writing by CLHIA, assignment of this Agreement will not release You from any prior outstanding obligation under this Agreement or allow You or Your assignee to expand the number of installations of the Software authorized under this Agreement. This Agreement is freely assignable by CLHIA and will inure to the benefit of CLHIA's successors and assigns. Any assignment in violation of this Section 13 is null and void.

14. Additional Terms. If any provision of this Agreement is found to be unenforceable, such term will be considered severable from the remaining terms, which will continue to be valid and enforceable. Under no circumstances will any other terms apply to this Agreement. No waiver of any of the terms or conditions of this Agreement will be binding for any purpose unless made in writing and signed by authorized representatives of both parties and any such waiver will be effective only in the specific instance and for the purpose given. No failure or delay on the part of either of the parties in exercising any right will operate as a waiver, nor will any single or partial exercise by the either of the parties of any right preclude any other or further exercise thereof or the exercise of any other right. All notices, consents and approvals under this Agreement must be delivered in writing by personal delivery, electronic transmission or certified mail, postage pre-paid, to the other party at its published address or at such other address as may be later designated by such party. Notices will be deemed to

have been received upon the date of receipt or, in the case of certified mailing, 2 days after deposit in the mail. This Agreement will be governed by the laws of Canada without regard to conflict of law principles. No agency, partnership, or joint venture is created by this Agreement. The parties are and remain at all times independent contractors and not agents or employees of the other party.