

H A R D E R

Software Development
and Consulting

S o f t w a r e E

CCDWS

Common Communication Driver for Web Services Technical Reference and Operating Manual

Revision 1.10
07-Aug-2012

Copyright © 2011-2012 HIEC / Harder Software Ltd.

Table of Contents

1.	Introduction	4
2.	Related Documents	4
3.	Run-Time Environment	4
3.1.	Operating Systems	4
3.2.	Required Run-Time Libraries	4
3.3.	Location of Run-Time Files	4
4.	Input and Output Files.....	5
4.1.	File Location	5
4.2.	Input File Generation	5
4.3.	Input File Format	5
4.4.	Output File Format	6
5.	Web Service Communications	6
5.1.	WSDL	6
5.2.	Sample Request Message Sent to Remote Server.....	6
5.3.	Sample Response Message Received from Remote Server.....	7
6.	Error Handling	7
7.	Configuration	7
7.1.	Configuration File Format	8
7.2.	Configuration Reference	8
8.	Automated Installation Instructions	15
8.1.	Before Installation.....	15
8.2.	Running the Setup Program	15
8.3.	Accepting the EULA and Reading Installation Notes	15
8.4.	Setup Type.....	15
8.5.	Destination Folder.....	15
8.6.	Actions Performed on Installation	15
8.7.	Configuration File Review and Edit	16
8.8.	Completion and Starting the CCDWS Service	16
8.9.	Additional Tasks Upon Successful Installation.....	16
9.	Manual Installation Instructions.....	16
9.1.	Before Installation.....	16
9.2.	Extracting Files	16
9.3.	Copying Certificate Files	17
9.4.	Updating the CCDWS Configuration File	17
9.5.	Updating the Existing CCD Configuration File.....	17
9.6.	Confirming CCDWS Operation.....	17
9.7.	Installing the CCDWS Service	18
10.	Controlling the CCDWS Service	18
10.1.	GUI Control	18
10.2.	Command Line Control	18
11.	Executing CCDWS from the Command Line.....	19
12.	Log Files	19
12.1.	Log File Format	19
12.2.	Configuring Log Filenames and Levels.....	19
12.3.	Log File Rotation	20

12.4.	Sample Log File with Notes.....	20
13.	Status Codes	22
14.	Internal Error Codes	23
15.	OpenSSL License	25
16.	gSOAP License	27

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 Main rather than Destination X.

1. Introduction

This document provides technical and operational information for the Common Communications Driver for Web Services (CCDWS), sponsored by the Health Industry Electronic Commerce (HIEC) associations, 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, polling for input request files. Upon discovery of an input file it will perform all tasks necessary to submit the claim and receive the adjudication response or acknowledgement in real time. 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.

2. Related Documents

To obtain copies of the following documents, contact a participating HIEC member.

CCDWS Requirements was authored by Alberta Blue Cross to provide the requirements and overall parameters for development of CCDWS.

CCDWS – Technical Design provides information on design and programming of the CCDWS application. This document was authored by Harder Software.

CCDWS – Test Plan can be used by software vendors and carriers to ensure that CCDWS is working as designed. This document was authored by Harder Software.

Point of Service (PoS) Application Services Transport Specification, Draft 1.9 defines much of the underlying transport specific specifications used for SOAP transactions. This document is authored by HIEC.

3. Run-Time Environment

3.1. Operating Systems

CCDWS is currently compiled to execute in recent versions of 32 or 64-bit Windows. It can be ported to Linux (various distributions), and to Mac OS X.

3.2. 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.

3.3. Location of Run-Time Files

CCDWS files can be installed in any local filesystem location in the Windows environment, and these locations can be controlled by the command-line parameters during installation and the configuration file options. However, the default values will allow CCDWS to run seamlessly alongside existing CCD installations.

The file locations are as follows:

`c:\ccd\ccdws.exe`: the CCDWS executable

c:\ccd\ccdws.ini: the CCDWS configuration file
c:\ccd\ccdws.log: the CCDWS log file
c:\ccd\<dest>\input.<ext>: input file containing the request payload
c:\ccd\<dest>\output.<ext>: structured output file corresponding to the input file
c:\ccd\<dest>\<acert.pem>: carrier's CA certificate file
c:\ccd\<dest>\<cert.pem>: client certificate for accessing the carrier

Where:

<dest> is the destination directory, referenced by the [Destination *<dest>*] directory in *ccdws.ini*. The value of *<dest>* will often be a carrier abbreviation. For instance ABC = Alberta Blue Cross.

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

<acert.pem> is the carrier's CA certificate filename as referenced in the configuration file.

<cert.pem> is the client certificate filename as referenced in the configuration file.

4. Input and Output Files

The CCDWS application communicates with the dental office software using file based inputs and outputs. This is consistent with the CCD application, allowing coexistence of CCD with CCDWS, and seamless integration of CCDWS within an existing dental software installation.

4.1. 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.

4.2. 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.

4.3. 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.

4.4. 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.

5. Web Service Communications

The CCDWS application is designed to support web service communications. 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 document entitled *CCDWS – Technical Design* contains the Web Services Design Language (WSDL) file used to define the web service call.

5.2. Sample Request Message Sent to Remote Server

The following message is generated by wrapping the CDAnet 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>
    </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. 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. 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 XYZ], 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] 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.

7.2. Configuration Reference

Configuration names and descriptions are shown below. See the section *Sample Configuration File* to view the 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: 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.

See Also: CertificateFile, CertificateKeyPassword, SSLVerify

CertificateFile

Sections: [Destination X]

Default Value: None

Required: No

Description: Specifies the file name for 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 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.

See Also: CACertificateFile, CertificateKeyPassword, SSLVerify

CertificateKeyPassword

Sections: [Destination X]

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

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

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

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

LogBackupExtension

Sections: [Main]

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]

Default Value: value of RootDir

Required: No

Description: Specifies the directory where log files are stored.

See Also: LogFile, LogLevel

LogFile

Sections: [Main]

Default Value: None

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]

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]

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.

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

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 uppcased. Additionally, for CDANET, it will also automatically append the version number to the *Format* node.

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.

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

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: ProxyServer

ReadWriteTimeout

Sections: [Destination X]

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

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

SSLVerify

Sections: [Destination X]

Default Value: 1

Required: No

Description: By default, if SSL/TLS encryption is used, CCDWS will verify the remote server based on the Certificate Authority certificate (CACertificateFile). Occasionally, changes on the server side may result in failure of the verification, so that the remote connection fails. 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

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

8. Automated Installation Instructions

These instructions assume that CCDWS will be installed in Windows, and that the default file locations are used. 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.

Additionally, the installation application will provide an opportunity to make configuration file changes prior to starting the installed CCDWS service. One possible change is the name of the client certificate file(s) for each destination. Default values will be provided, but each software vendor is provided with a differently named certificate, so unless this certificate is given the default name, it may need to be altered.

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.

8.2. Running the Setup Program

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

8.3. Accepting the EULA and Reading Installation Notes

When installing for the first time, you will be required to review and accept the End User License Agreement (EULA) before continuing. Once accepted, recent installation notes will be displayed. Please read these notes carefully before continuing.

8.4. 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.

8.5. 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.

8.6. 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.

8.7. 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.

8.8. 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.

8.9. 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 that were installed, then they will need to be placed in the appropriate destination subdirectory. The names of these certificates should match those in the configuration file. Note that the CCDWS service will not need to be restarted after these files are placed. It will only need to be restarted if the configuration file is subsequently changed.

9. Manual Installation Instructions

These instructions assume that CCDWS will be installed in Windows, and 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*). Locations are provided in the *Run-Time Environment* section.

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.

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. You should see a single directory `ccd` within that archive. Drag this with your mouse to drive C.

The directory `c:\ccd` may already exist. If so, you will be prompted to merge with the existing location. Select *Yes*. When prompted whether to replace any files, also select *Yes*.

9.3. Copying Certificate Files

Each destination administrator should have provided a CA certificate file and (optionally) a certificate/key pair. Copy those files to the appropriate location. For instance, if Alberta Blue Cross certificates are available, copy these to the directory `c:\ccd\abc`. If the directory does not exist, you may first need to create it. Repeat this for all destinations. The destination CA certificate may already be installed there as part of the extraction process. All CA certificates and certificate/key pairs should have `.pem` extensions.

9.4. Updating the CCDWS Configuration File

The 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. The destination section name is case insensitive.

9.5. 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.

9.6. 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:

9.7. Installing the CCDWS Service

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*.

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
```

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, open a command prompt and enter the following commands (assuming CCDWS is installed in the default location):

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

The application will continue to run in the foreground until the command prompt is closed, or *Ctrl-C* is pressed.

In this mode 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 (c:/ccd).

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*.

Log levels may be set individually for the main thread and each listener thread, or combined in the global section of the configuration file. 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
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/CDANE
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://hiec.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.
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 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	No such file or directory.	The specified file or directory does not exist or cannot be found. This message can occur whenever a specified file does not exist or a component of a path does not specify an existing directory.
8	Out of memory.	Not enough memory on CCDWS host, or there is a memory leak. Try rebooting workstation. Contact support personnel if the problem persists.
13	Permission 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 of its read and write locations.
21	Is a directory.	The file attempting to be accessed is a directory, not a regular file. Check the configuration option for the file being used, and check the file system to ensure that a directory of the same name does not exist.
24	Too many open files.	No more file descriptors are available, so no more files can be opened.
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.

Code	Message	Description
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.
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.

Code	Message	Description
7407	HTTP: Proxy Authentication. Required.	Ensure that the ProxyPassword configuration option is 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. 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 (eay@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.]

16. 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>.